UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

APLICAÇÃO DA INTELIGÊNCIA ARTIFICIAL NA ENGENHARIA RODOVIÁRIA

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA

PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

ILSON WILMAR RODRIGUES FILHO

FLORIANÓPOLIS

SANTA CATARINA - BRASIL

MARÇO/90

APLICAÇÃO DA INTELIGÊNCIA ARTIFICIAL NA ENGENHARIA RODOVIÁRIA

ILSON WILMAR RODRIGUES FILHO

Esta dissertação foi julgada adequada para a obtenção do título de

"MESTRE EM ENGENHARIA"

Especialidade Engenharia de Produção e aprovada em sua forma final pelo Programa de Pós-Graduação.

Banca Examinadora:

Prof.: Ricardo Miranda Barcia, Ph.D.
- Coordenador -

Prof.: Ricardo Miranda Barcia, Ph.D.
- Orientador -

Prof.: Renato Antônio Rabuske, Dr.Sc.
- Co-orientador -

Prof.: Amir Mattar Valente, M.Sc.

Prof.: Neri dos Santos, Dr.Eng.

Para Dóti, Laura e Vírginia.

AGRADECIMENTOS

Gostaria de expressar meus agradecimentos:

- Ao Prof. Ricardo Miranda Barcia pela sugestão do assunto para dissertação, orientação geral do trabalho e pelo apoio oferecido durante o curso;

- Ao Prof. Renato Antônio Rabuske, pela co-orientação e pela apresentação de críticas ao trabalho;

- À profa. Márcia Aguiar Rabuske sobre as longas conversas sobre Prolog;

- Ao Névio e à Sandra pelo apoio logístico;

- Aos colegas da Pós-graduação, especialmente ao Fernando Gauthier, pelas discussões que permitiram avanços no aprendizado de Prolog;

- Aos funcionários da Pós-Graduação e do Departamento de Engenharia de Produção (Zelita, Margarete, Aldanei e Namir);

- À CAPES pelo apoio financeiro;

- Aos colegas do Departamento de Estradas de Rodagem de Santa Catarina, Engenheiros Nei Damo, Edgar Roman e Nolli, que forneceram o conhecimento para a construção do sistema CONSER;

- À todos àqueles que de uma forma ou outra contribuiram para o desenvolvimento deste trabalho.

RESUMO

Os engenheiros rodoviários (bem como profissionais de outras áreas de conhecimento) usam experiência, bom senso, analogia, intuição e raciocínio formal para resolver problemas. Estes processos de obtenção de conhecimento, que denominamos de processos heurísticos, são extremamente difíceis de serem armazenados em programas convencionais de computador. No entanto, técnicas desenvolvidas a partir da década de 50, que constituem uma área da Computação denominada de Inteligência Artificial, permitem a construção de Sistemas Especialistas que incorporam estas heurísticas e vão constituir uma ferramenta que reproduz o processo de resolução de problemas usado pelo especialista humano. Esta é a proposta de CONSER, um sistema especialista para apoio à escolha de estratégias de conservação de rodovias não pavimentadas.

ABSTRACT

Highway Engineers (as well as professional in another areas) use their experience, common sense, anologies, intuitive and formal thinking for solving problems. These processes for obtaining knowledge, called heuristics are not easily transferable to conventional computer programs. However by using a new branch of knowledge named Artificial Intelligence it is possible to build systems which, incorporating these heuristics, became a tool for reproducing the solution process used by human beings.

This work proposal is CONSER, an Expert System which helps to decides among alternativers for (conservação de rodovias não pavimentadas).

SUMÁRIO

1. INTRODUÇÃO

Os profissionais que trabalham na engenharia rodoviária, após muitos anos de experiência, adquirem um conhecimento que consiste basicamente de regras práticas, não encontradas nos livros e nos manuais, denominadas de heurísticas. São exatamente estas heurísticas que permitem aos engenheiros resolverem problemas, reconhecer rapidamente qual a melhor estratégia para resolvê-los e manipular com dados incompletos, inexatos ou mesmo errados. Isto ocorre por exemplo na construção e na conservação de rodovias.

Na conservação de rodovias, os engenheiros encarregados usam dos recursos disponíveis (não raro, insuficientes), às vezes com alguma engenhosidade, para mantê-las em condições de tráfego. Adotam-se técnicas localizadas (regionalizadas), que dependem do tipo de solo, clima, relevo, etc, para as quais a vivência por longo tempo numa determinada região, faz com que o engenheiro responsável pela conservação, venha a adquirir e desenvolver. Poderia, mesmo não sendo sua função, acompanhar o funcionamento da rodovia observando e anotando com detalhamento os problemas que surgissem com o tempo e uso. Isto permitiria a obtenção de parâmetros novos ou atualizados para aplicar em novos projetos ou mesmo a determinação de novos critérios para futuros projetos.

Temos assim um processo contínuo de aquisição de conhecimento que não é preservado, tendo em vista a rotatividade

dos especialistas pelas várias regiões e nos vários tipos de atividades relacionadas com estradas de rodagem.

Adquirir e reproduzir este tipo de conhecimento é a tarefa principal de sistemas especialistas, uma das aplicações da Inteligência Artificial.

Técnicas de Inteligência Artificial poderiam ser aplicadas na engenharia rodoviária. Este trabalho pretende mostrar isto, aplicando estas técnicas na conservação de rodovias, objetivando o armazenamento do conhecimento empírico de engenheiros que trabalham nesta área no Departamento de Estradas de Rodagem de Santa Catarina, DER/SC, cuja experiência, considerando-se que ela deva pertencer àquele órgão rodoviário, fique armazenada para posterior consulta. "Algumas organizações vêem sistemas especialistas como uma maneira de coletar e preservar a memória institucional garantindo-a contra a rotatividade dos especialistas humanos que podem se retirar das organizações, adoecer ou mesmo falecer" [Genaro,86]. Apresentamos no capítulo 4 deste trabalho, o programa CONSER que dá conselhos sobre estratégias de conservação, como o faria um especialista.

O conhecimento especializado e acumulado de anos de experiência na conservação rodoviária, poderia ser levado a órgãos onde os engenheiros rodoviários não estão disponíveis, tais como prefeituras de pequenas cidades. Cursos de treinamento poderiam ser oferecidos aos funcionários destas prefeituras e os sistemas

especialistas poderiam ser úteis nestes cursos.

Sistemas Especialistas poderiam assim ser usados na resolução de problemas de rotina da engenharia rodoviária, especificamente na área de conservação.

O propósito desta dissertação não é apresentar mais um sistema especialista, mas, mostrar a validade do uso de técnicas de Inteligência Artificial na engenharia rodoviária. A construção de um sistema especialista demanda muitos homens/hora de trabalho. O tempo necessário para construi-lo vai depender da complexidade do problema e do número de profissionais disponíveis para a sua construção. Segundo Genaro [Genaro,86], uma tarefa de moderada dificuldade, poderá ser resolvida entre seis a dezoito meses, utilizando-se duas a quatro pessoas, em dedicação exclusiva. Uma tarefa difícil pode demorar de um a tres anos com envolvimento de tres a quatro pessoas em tempo integral.

2. PEQUENO HISTÓRICO DA INTELIGÊNCIA ARTIFICIAL

2.1 Antecendentes da Inteligência Artificial

Desde a antiguidade o homem tem lidado com a idéia de dotar dispositivos artificiais, sejam metálicos, de madeira ou de marfim, de movimentos e inteligência. Estátuas com movimentos já eram construídas por gregos, judeus e outros povos da época. A mitologia grega é rica em referências a andróides capazes de trabalhar e falar, mesas móveis para servir aos deuses no Olimpo, etc.

Na ilha de Chipre vivia um escultor, assim reza a lenda, de nome Pigmalião. Nesta mesma ilha, viviam também as Propétidas, filhas de Amatonte. Elas haviam se prostituído, e devido a este fato, Pigmalião tomado de ódio pelas mulheres, tornou-se celibatário. Afrodite (ou Venus, para os romanos), deu vida a uma das estátuas de Pigmalião. Era uma bela estátua, toda de marfim. A estátua era tão bonita que Pigmalião havia se apaixonado por ela. A estátua tornou-se uma bela mulher que casou-se com Pigmalião e com ele teve um filho, chamado Pafo.

A tradição judaica fala de Golem, um autômato antropomórfico, que é animado por intervenção divina.

Roger Bacon (1214-1294) teria construído uma cabeça que falava. Leonardo da Vinci (1452-1519), em honra a Luís XII,

construiu um leão mecânico que se movia.

Nos séculos XVII e XVIII eram famosos os tocadores de flauta automáticos, e o brinquedo que escrevia (Escriba), o que desenhava (Desenhista), e o Músico que podia tocar um órgão em miniatura. Estes tres autômatos (Escriba, Desenhista e Músico) estão guardados no Museu de Arte e História, em Neuchatel, na Suiça.

Durante o século XIX ocorre um grande avanço da mecânica. Neste século surge uma variedade de máquinas falantes, jogadores de xadrez, etc.

No século XX, ocorre o auge da cibernética. Em 1912, o cientista espanhol Leonardo Torres y Quevedo, presidente da Academia de Ciência de Madrid, construiu um autômato que jogando xadrez com o rei branco e um peão, dava xeque-mate no rei preto.

Em 1932, robôs que falavam, fumavam charutos e pressupostamente liam jornais, foram exibidos na Rádio Londres. Em 1939, na Feira Mundial de Nova Iorque, é apresentado o Elektro, que era capaz de realizar vinte e seis movimentos diferentes e de obedecer comandos verbais.

2.2 Início da Inteligência Artificial

O termo Inteligência Artificial (IA) foi criado por John McCarthy, da Universidade de Stanford, em 1956, para batizar a área da ciência da computação que iniciava-se em meados dos anos 50, nos Estados Unidos.

MacCarthy, convocou naquele ano, alguns pesquisadores que estavam trabalhando nesta área, que se delineava então como um ramo da ciência da computação, para um simpósio em Darmonth, no verão. Neste simpósio compareceram Allen Newell, Herbert Simon e Marvin Minsky, que juntamente com John McCarthy são considerados os pioneiros da Inteligência Artificial.

Em 1956, Allen Newell, J.C.Shaw e H.A.Simon pesquisavam técnicas de resolução de problemas semelhantes à forma como as pessoas fazem, com uso de heurísticas. Este modo apresenta certas vantagens em relação ao procedimento algorítmico, quando em face a uma variedade muito grande de caminhos alternativos a seguir em busca da solução. Newell e Simon apresentaram no simpósio, o programa "Logic Theorist" que tinha por objetivo possibilitar ao computador jogar xadrez, provar teoremas matemáticos e descobrir leis a partir de dados, desenvolvido no Instituto de Tecnologia Carnegie, em Pittisburg, conhecido hoje como Universidade Carnegie-Mellon. J. C. Shaw era da Rand Corporation. Os três, haviam desenvolvido o IPL - Information Processing Language, linguagem de manipulação simbólica, com a qual construíram o Logic

Theorist.

O **Logic Theorist** provou vários teoremas do "Principia Mathematica" de Alfred North Whitehead, matemático e filósofo inglês e Bertrand A. W. Russel, filósofo e matemático, nascido no País de Gales, obra esta onde os autores deduzem a partir de um número reduzido de axiomas, as partes principais da teoria dos conjuntos e dos números reais. Esta obra havia sido escrita entre 1900 e 1910. O **Logic Theorist** foi pioneiro no uso de heurísticas em programas [Barr,81].

Newell, Shaw e Simon, aperfeiçoando o Logic Theorist, constroem o GPS - General Problem Solver. O objetivo do GPS era decifrar quebra-cabeças, provar teoremas e encontrar integrais indefinidas. Apesar de que muitos problemas propostos ao GPS, não conseguiam ser resolvidos, este programa teve uma grande contribuição ao desenvolvimento da IA: para a sua elaboração foi desenvolvida toda uma metodologia para resolução de problemas com métodos e processos supostamente iguais aos processos humanos. O GPS separa os métodos de resolução de problemas, do conhecimento específico sobre o mesmo [Barr,81]. Nisto reside a sua importância, não apenas do ponto de vista histórico. Foi, portanto, um marco na História da Inteligência Artificial. Um de seus criadores, Herbert Simon, recebeu o prêmio Nobel de Economia, em 1978.

Em 1958, John McCarthy cria a linguagem LISP - List Processing, a partir do IPL, cuja característica fundamental é a facilidade de manipulação de listas. LISP veio a se tornar uma das linguagens preferidas na programação de IA.

Na década de 60 e início de 70, cientistas em informática, juntaram-se a psicólogos, filósofos e linguistas, criando as tres seguintes áreas em Inteligência Artificial:

- sistemas especialistas;
- reconhecimento de imagens;
- processamento de linguagem natural.

A partir daí as técnicas de Inteligência Artificial (IA) foram se desenvolvendo, principalmente nas Universidades de Stanford, Massachusets Institute of Technology (MIT) e Carnegie-Mellon University.

A Universidade de Stanford inicia em 1965 o seu Projeto de Programação Heurística, desenvolvendo o DENDRAL (Joshua Lederberg e Edward Feigenbaum), programa para fazer análises químicas, que teve bom desempenho.

Em 1966, Joseph Weisenbaum do MIT publica o programa ELIZA ("ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine"). ELIZA simulava o comportamento de um terapeuta rogeriano (Psicanalista que segue a linha terapêutica de Karl Rogers). Nesta linha de análise, o

psicanalista toma uma posição passiva, retrucando as observações do paciente ao invés de comandar a conversação.

No início dos anos 70, surgiu o mais famoso sistema especialista, chamado MYCIN, que foi desenvolvido na Universidade de Stanford e foi criado para ajudar médicos na consulta de infecções de bactérias no sangue e meningite (infecções que provocam a inflamação das membranas que envolvem o cérebro e a medula). Estas doenças infecciosas manifestam-se com muita frequencia durante a hospitalização do paciente. O MYCIN foi projetado para ajudar os médicos internos. Foi o primeiro sistema especialista a equiparar-se a um especialista humano, e a dar ao usuário explicações do seu raciocínio.

Em 1972, em Marseille, França, foi criado a linguagem Prolog, por Alain Colmerauer. Até então só se dispunha para escrever programas em IA, da linguagem Lisp.

Na década de 80 renasce a pesquisa sobre redes neurais, cujos fundamentos teóricos haviam sido lançados em 1943 por W. S. McCulloch e S. A. Pitts.

3. CONCEITOS DE INTELIGÊNCIA ARTIFICIAL

3.1 Introdução

Antes de iniciarmos de fato o estudo de Inteligência Artificial, vamos fazer uma pequena discussão sobre o que seja inteligência e programas inteligentes. Para definirmos o que é um programa inteligente, uma pequena preleção sobre inteligência se faz necessário. O Novo Dicionário Aurélio dá a seguinte definição de inteligência: "1. Faculdade de aprender, apreender ou compreender, percepção, apreensão, intelecto, intelectualidade. 2. Qualidade ou capacidade de compreender e adaptar-se facilmente; capacidade, penetração, agudeza, perspicácia..."(Aurélio Buarque de Holanda Ferreira, Novo Dicionário da Língua Portuguesa, Editora Nova Fronteira, 1a. edição - 10a. impressão, pág. 774). O Pequeno Dicionário Enciclopédico Koogan Larousse, da Editora Larousse do Brasil, edição de 1979, define inteligência como: "Faculdade de conhecer, de compreender: a inteligência distingue o homem do animal...". Por esta segunda definição não poderíamos caracterizar um programa como sendo inteligente, se inteligência for uma qualidade restrita aos seres humanos.

Se um computador for capaz de executar tarefas que sejam consideradas próprias para seres humanos, tais como fazer contas, ler instruções em linguagem natural ou mesmo entender ordens verbais, podemos dizer que este computador também é inteligente?

Muitos estudiosos descrevem inteligência como a capacidade de resolver problemas. J. Chaplin a define como "Capacidade de resolver problemas frente a novas situações. Capacidade de utilizar efetivamente conceitos abstratos". Henry E. Garrett, professor de Psicologia na Columbia University, no livro "Psicologia", diz que "O Emprego da inteligência para a solução de problemas requer tanto capacidade como velocidade" . Os computadores atuais tem capacidade de memória de megabytes e podem executar milhões de cálculos por segundo. Alguns animais podem resolver alguns tipos de problemas. Encontram mesmo soluções com situações novas que surjam. Logo, animais também podem resolver problemas. Quanto a computadores, isto é indiscutível, além do que computadores podem resolver problemas muito mais rapidamente que os seres humanos. Usando técnicas de IA, as situações não necessitam ser previstas antecipadamente pelo programador.

Os psicólogos costumam usar testes de inteligência para poder caracterizá-la. Thurstone (1887-1955), psicólogo norte-americano que se celebrizou por seu trabalho pioneiro na área dos testes de inteligência, e dentre suas principais obras está a "The Nature of Intelligence", propôs sete fatores, independentes entre si, para caracterizar inteligência:

(1) habilidade para definir e compreender palavras;

(2) habilidade de encontrar palavras rapidamente para improvisar um discurso ou resolver uma charada;

(3) habilidade para resolver problemas aritméticos;

(4) habilidade para desenhar de memória ou visualizar relações;

(5) habilidade para memorizar e relembrar;

(6) habilidade para apreender pormenores visuais e perceber diferenças e semelhanças;

(7) habilidade para descobrir as leis e os princípios que regem as coisas ou formar conceitos para resolver problemas (Enciclopédia Barsa, Encyclopaedia Britannica do Brasil Publicações Ltda, Volume 9, 1984).

Os computadores satisfazem bem ou mal a todos estes requisitos.

Quaisquer definições que tentemos tomar sobre inteligência, de alguma forma poderemos enquadrar nelas, programas de computador, argumentando assim que eles são inteligentes.

Vê-se assim que o conceito de inteligência é bastante difícil de ser estabelecido, e daí, talvez, tenhamos alguma dificuldade em definir inteligência artificial. Isto ocorre pelo fato da própria inteligência não estar bem definida e pelo fato de que se reluta em admitir que dispositivos artificiais possam

desenvolver atributos os quais caracterizam os seres humanos.

Mas, vamos definir programa inteligente como aquele que tem um comportamento similar ao de um ser humano na resolução de problemas, independente da forma que o programa resolve o problema, desde que obtenha um resultado idêntico ao obtido por um ser humano.

3.2 Definição

Inteligência Artificial é a ciência de fazer o computador realizar tarefas que até agora, somente as pessoas podiam realizar e somente pessoas muito especializadas podiam realizar bem [Rich,88]. "É o ramo da ciência da computação que objetiva desenvolver sistemas de computador que exibam as características que nós associamos com a inteligência e comportamento humano, tais como: entender linguagens, raciocinar, resolver problemas, etc [Barr,81].

Segundo Araribóia [Araribóia,88], um computador é inteligente se possui qualquer uma das habilidades mentais que fazem uma pessoa ser considerada inteligente. Entre estas habilidades citam-se:

a) capacidade de raciocinar e de realizar inferências;

b) capacidade de resolver problemas;

c) capacidade de acumular e de usar conhecimentos;

d) capacidade de falar linguas humanas;

e) capacidade de planejar as próprias ações e de prever o resultado delas;

f) capacidade de aprender com a experiência;

g) capacidade de ver, de ouvir e de interpretar estímulos sensoriais.

Algumas destas habilidades já foram incorporadas a computadores.

Pode-se concluir que para uma máquina ser inteligente, ela deve ter conhecimento que possa se explorado. E este conhecimento deve ser tal que possa ser utilizado mesmo em situações em que ele esteja incompleto, Ele deve ser facilmente modificado e facilmente acessado.

Os trabalhos sobre Inteligência Artificial iniciaram ainda no tempo dos computadores que utilizavam válvulas. Alan Turing foi um dos pioneiros no campo da IA. Ele imaginou um teste para determinar se uma máquina é inteligente, que tem sido desde então chamado de Teste de Turing, e que consta do seguinte: em duas salas separadas coloca-se um computador e uma pessoa para serem interrogados por uma outra pessoa que não sabe em qual sala está o computador. Caso esta pessoa não consiga descobrir onde está o computador, ele (o computador) será considerado inteligente. É uma adaptação de uma brincadeira: adivinhar se quem

responde é homem ou mulher.

Outras expressões já foram sugeridas para substituir Inteligência Artificial, haja visto que esta forma tem suscitado muita discussão. Na Universidade de Carnegie-Mellon, foram sugeridos "processamento complexo de informação" e "simulação de processos cognitivos".

3.3 Representação do Conhecimento

Em qualquer sistema especialista (SE), uma questão importante a ser tratada é a forma como o conhecimento é representado. O conjunto dos fatos e regras no SE constitui a sua base de conhecimento. Estes fatos são obtidos em bibliografia e/ou entre peritos humanos em áreas específicas das atividades humanas. Precisamos ter mecanismos para armazenar e manipular este conhecimento. O conjunto destes mecanismos constituem a Representação do Conhecimento. Estes mecanismos devem levar em conta as particularidades da área de aplicação do SE. Com eles se escreverá a base de conhecimento e sobre esta base atuará o que se chama de máquina de inferência, que é a parte do programa que acessa o conhecimento que está relacionado com o problema diretamente da base, ou faz inferências sobre este conhecimento para produzir a solução e/ou adquirir mais conhecimento. A máquina de inferência poderá executar, em alguns sistemas, ainda tarefas cognitivas tais como ver, manipular robôs e tomar decisões.

Várias maneiras de representar conhecimento tem sido utilizadas em Inteligência Artificial. As principais são as seguintes:

a) Representação em lógica;
b) Sistemas de produção;
c) Redes semânticas;
d) Frames.

3.3.1 Representação em lógica

A lógica tem uma importância muito grande na Inteligência Artificial e em especial em Prolog. O primeiro estudo sobre lógica tem sido atribuido ao filósofo grego Aristóteles (384-322 AC). Ele desenvolveu boa parte da teoria que tem sido denominada de lógica clássica. Ela trabalha essencialmente com assertivas, chamadas proposições que podem ser verdadeiras ou falsas.

Um grupo de proposições constituem um argumento. Seja, por exemplo o seguinte argumento apresentado por Schildt [Schildt,87]:

João é um homem.
Todos os homens foram meninos.
Portanto, João foi um menino.

Proposição é um enunciado que admite valor lógico. Poderíamos dizer que proposição é o significado de uma sentença. Por exemplo:

> João ama Maria.
> Maria é amada por João.

são duas sentenças diferentes, mas que tem o mesmo significado, ou seja, tem a mesma proposição.

Seja o seguinte argumento com tres proposições apresentadas por Copi [Copi,78]:

> Tudo o que é predeterminado, é necessário.
> Todo evento é predeterminado.
> Logo, todo evento é necessário.

Destas três proposições, duas são premissas (as duas primeiras) e uma é a conclusão (a terceira).

Uma proposição pode ser premissa num argumento e conclusão em outro. Por exemplo:

> Todo evento causado por outros eventos é predeterminado.
> Todo evento é causado por outros eventos.
> Logo, todo evento é prederminado.[Copi,78]

A proposição "Todo evento é predeterminado" é conclusão no último argumento e é premissa no anterior a ele.

Do que vimos, pode-se concluir que a lógica clássica é simplesmente uma formalização do senso comum. E, é baseada na linguagem natural. Os argumentos são formulados em linguagem natural (no nosso caso, em portugues), e, são com frequência de difícil avaliação por causa da natureza vaga e imprecisa das palavras usadas, de estilos metafóricos, confusos, etc. Criou-se assim a lógica simbólica. Os símbolos especiais da lógica simbólica permitem expor com mais clareza, as estruturas lógicas de proposições e argumentos, cujas formas poderiam ser obscurecidas pela pouca maleabilidade da linguagem corrente [Copi,78].

Seja, por exemplo, as seguintes premissas de um argumento apresentado por Genaro [Genaro,86]:

1. Se for inverno em Paris, será verão em Brasília

2. Se for verão em Brasília, então está chovendo

3. Agora é inverno em Paris

Estas premissas estão expressas em linguagem corrente. Utilizando símbolos para representá-los temos:

P : inverno em Paris

Q : verão em Brasília

R : está chovendo em Brasília

Temos assim:

1. P ---> Q
2. Q ---> R
3. P

Destas tres premissas podemos deduzir a conclusão:

4. R

Esta parte da lógica simbólica, que trabalha com proposições utilizando símbolos para representar estas proposições, é denominada lógica proposicional (ou ainda: cálculo dos enunciados, cálculo sentencial ou cálculo proposicional [Cerqueira,79]). Ela usa operadores para unir estas proposições, tais como:

e : ^
ou : v
não : ~ ou ¬
implica : ----->
se e sòmente se : <----->

O poder de representação da lógica proposicional é porém limitado. Ela não é suficiente para expressar substantivos,

pronomes, verbos, adjetivos ou advérbios. Só se utiliza símbolos para representar sentenças completas (as proposições). É preciso pois introduzir uma forma de obter esta representação. A lógica dos predicados ou cálculo dos predicados permitirá a formulação de afirmações que seriam impossíveis na lógica proposicional. Permitirá ainda o uso de funções e variáveis.

Na linguagem do cálculo dos predicados, um enunciado do tipo:

"o trecho é longo"

poderia ser encarado como um enunciado simples simbolizável, usando-se uma variável como por exemplo 'T'. Prestando-se mais atenção ao enunciado, podemos observar que ele tem duas partes distintas do ponto de vista gramatical: "o trecho" que é o sujeito da frase, e um predicado "longo" que indica um atributo do trecho.

No cálculo dos predicados assumem enorme relevância as expressões que descrevem ou nomeiam indivíduos [Cerqueira,79].

Alain Colmerauer foi um dos primeiros pesquisadores a usar o cálculo de predicados para programar. Predicado é um meio útil de representar e manipular alguns dos tipos de conhecimentos necessários em sistemas de IA. Predicados são declarações a respeito de objetos em si, ou sobre relações dos objetos entre si.

Usa-se letras maiúsculas para representar os predicados. Para representar constantes usa-se letras do início do alfabeto. Dado o enunciado:

o trecho é longo

podemos empregar 'L' no lugar do predicado "é longo" e 't' no lugar de "trecho", obtendo:

Lt, que se lê: L de t

O enunciado "o trecho é plano" pode ser representado por:

Pt, que se lê: P de t

onde 't' corresponde a "trecho" e 'P' a "é plano".

A negação de tal enunciado será:

~ Pt --> que se lê: não-P de t

Os dois enunciados simples, dados acima, poderiam ser representados com apenas um enunciado composto:

Lt ^ Pt --> que se lê: L de t e P de t

Esta estruturação do Cálculo dos Predicados foi dada pelo filósofo alemão Gottlob Frege, no fim do século 19. Um matemático também alemão chamado Alfred Horn, restruturou o cálculo dos predicados. Na estrutura proposta por Horn os enunciados "o trecho é longo" e "o trecho é plano" podem ser escritos como:

```
qualificador(trecho,comprimento,longo)
qualificador(trecho,declividade,plano)
```

onde temos o objeto "trecho" com os qualificadores "comprimento" e "declividade" que tem os valores "longo" e "plano" respectivamente.

Este tipo de expressão foi chamada de sentença de Horn. Com esta forma as expressões lógicas podem ser programadas eficientemente em computadores, conforme descobriram isto pioneiramente Alain Colmerauer e seu aluno Philippe Roussel, em 1973. Ao sistema computacional capaz de fazer deduções a partir do Cálculo dos Predicados sob a forma de sentenças de Horn deram o nome de PROLOG. E, os programas deveriam conter apenas sentenças de Horn. Mas, a princípio o Prolog apresentou algumas dificuldades. "O PROLOG inventado por Colmerauer e Roussel não era uma verdadeira linguagem de programação pois não conseguia executar certos tipos de programas... O primeiro PROLOG verdadeiro só foi construído em 1979 por David Warren, Luís Muniz Pereira e Fernando Pereira" [Araribóia,88].

Vejamos o seguinte conjunto de frases (fatos) apresentado por Rich [Rich,88]:

1. Marco era um homem.
2. Marco era um pompeano.
3. César era um soberano.

Estes fatos podem ser representados da seguinte forma, usando sentenças de Horn:

1. homem(Marco).
2. pompeano(Marco).
3. soberano(César).

Nos exemplos anteriores, os predicados tem apenas um **argumento** cada (os elementos que estão dentro dos parenteses). Mas, poderiam ter qualquer número deles. Ao número de argumentos chamamos de **aridade** do predicado.

O cálculo dos predicados usa ainda o que se chama de quantificadores. Exemplos:

- para todo;
- existe.

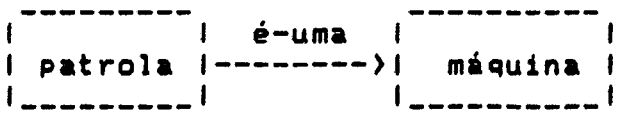
Vamos supor a seguinte frase: "Todo homem é mortal". Ela pode ser expressa no cálculo de predicados da seguinte forma:

para-todo(X): se homem(X) então mortal(X).

3.3.2 Redes Semânticas

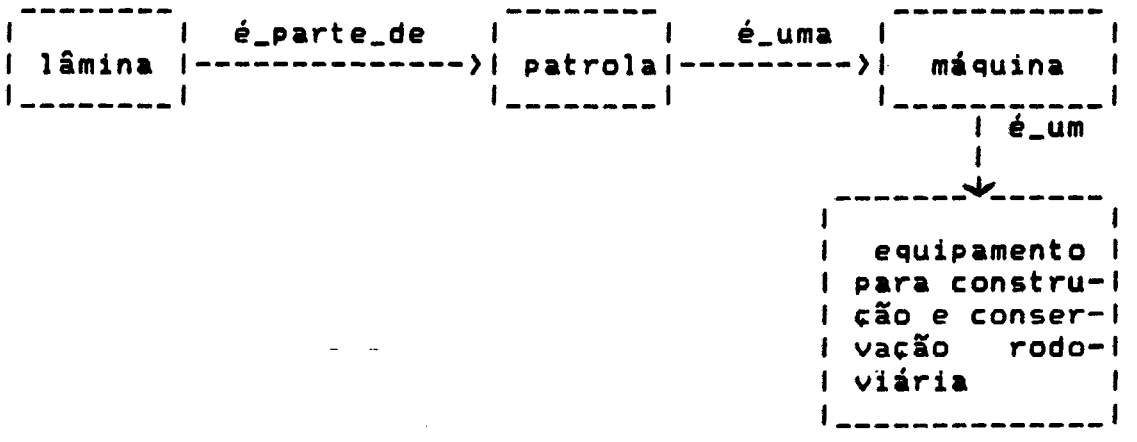
A forma de representação de conhecimento conhecida como rede semântica é uma das primeiras na Inteligência Artificial [Harmon,88]. Redes Semânticas são estruturas formadas por nós, conectados entre si por arcos rotulados.

Os nós geralmente representam objetos. Mas, também podem representar conceitos, eventos, ações ou situações de um determinado domínio. Os arcos (ou elos) representam as relações entre os nós. Um tipo de arco frequentemente utilizado entre um nó representando uma classe de objetos e um nó exemplo dessa classe, é o arco **é_um(a)**:

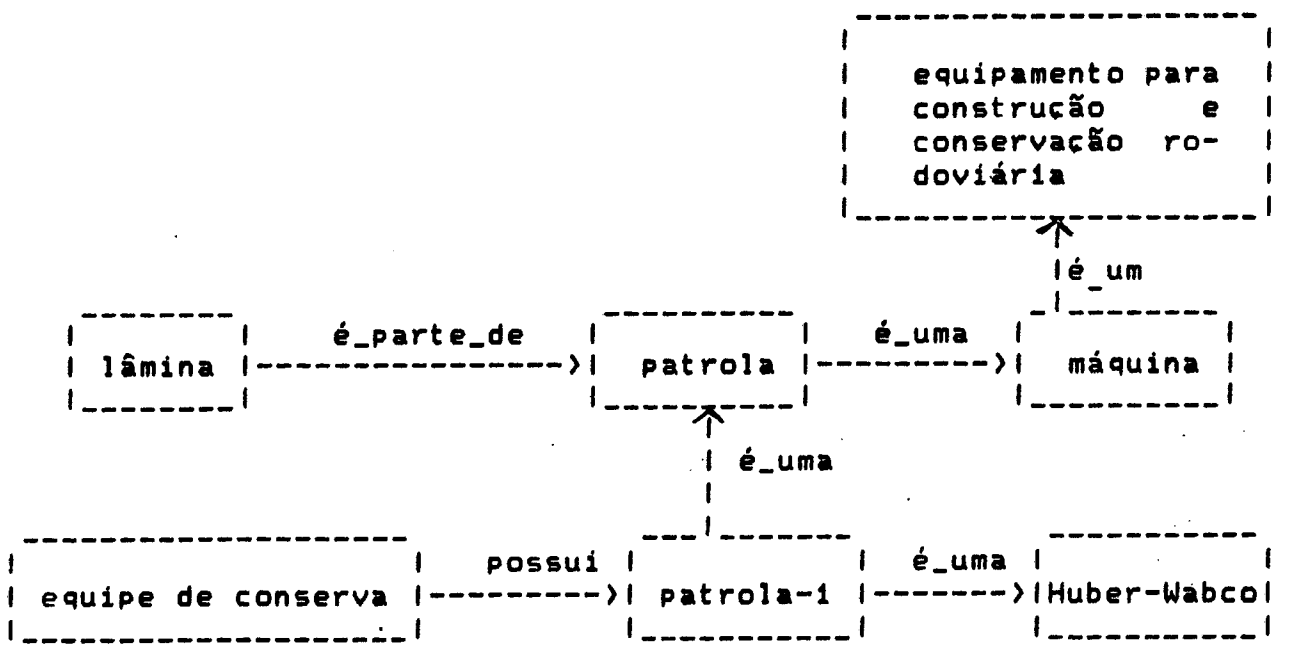


onde máquina é uma classe de objetos e patrola é um objeto pertencente a esta classe.

Esta forma de representação do conhecimento tem uma grande vantagem: a flexibilidade. Outros fatos podem ser acrescentados à rede, à medida que se precise. Por exemplo, os fatos: "toda patrola tem lâmina" e "uma patrola é uma máquina para construção e conservação rodoviária" podem ser acrescentados:



Se quizermos representar o fato: "a equipe de conserva tem uma patrola", a rede passará a ter a seguinte configuração:



No nosso caso "patrola-1" é um caso particular da classe "patrola", que tem características particulares que talvez não sejam comuns a todas as patrolas. A marca da "patrola-1" é "Huber-Wabco". Outras patrolas podem ser de outras marcas.

Uma característica das redes semânticas é a hereditariedade, ou seja, os nós herdam as características de outros nós relacionados com ele. É causado pela relação é_um(a). Se "patrola-1" é_uma "patrola", então "patrola-1" tem as características de "patrola", o que parece ser bem óbvio. A hereditariedade poderia ser realizada por regras, como as seguintes escritas em Prolog:

```
é_um_tipo_de(X,Z):-
                    é_um(X,Z).

é_um_tipo_de(X,Z):-
                    é_um(X,Y),
                    é_um_tipo_de(Y,Z).
```

As relações "é_um" devem ser acrescentadas ao programa da seguinte forma:

```
é_um(patrola_i,patrola).
é_um(patrola,máquina).
é_um(máquina,equipamento_conservação_construção).
```

Se perguntarmos ao programa o que é "patrola_i" através da consulta:

```
é_um_tipo de(patrola_i,Tipo)
```

obteríamos como resposta:

```
Tipo = patrola
Tipo = máquina
Tipo = equipamento_conservação_construção),
```

onde a informação de que patrola é uma máquina e um equipamento de conservação e construção foi obtida por hereditariedade.

3.3.3 Frames

Frames (que tem sido traduzido para o portugues como **quadros** ou **estantes**) são descrições estruturadas de situações estereotipadas, ou seja, de situações que não mudam, que são sempre iguais. Por exemplo, se vamos visitar pela primeira vez uma determinada pedreira, esperamos encontrar britadores, correias transportadoras, peneiras, pilhas de brita de vários diâmetros, poeira no ar, muito barulho, etc, baseado na experiência obtida em outras pedreiras. Nós temos assim uma expectativa do que deveremos ver e notaremos que algo não estará correto se não houver peneiras na pedreira: ou ela está desativada ou está ainda em instalação e as peneiras ainda não foram montadas, etc.

Os frames consistem de uma coleção de **slots** (prateleiras, nichos ou escaninhos), onde se descrevem aspectos dos objetos. Associados a cada **slot** poderá haver:

a) valores de atributos dos objetos;

b) restrições que estabelecem os tipos de valores possíveis ou o valor máximo desses valores;

c) procedimentos que permitam calcular a informação desejada.

Um exemplo ajudará a entender melhor a representação por frames. Um frame para o conceito genérico de bull-dozer deve ter informações sobre o proprietário, peso, capacidade do tanque de combustível, etc. Cada uma destas informações é guardada num slot:

Frame: BULL-DOZER

slot	Conteúdo
Marca	Sequencia de letras
Proprietário	Particular ou Empresa Pública
Peso	Um número real
Capacidade do Tanque	Um número real

3.4 Processos de Busca

Na resolução de um problema aplicando técnicas de IA, devemos ter em consideração três coisas:

(1) um sistema de representação do problema;

(2) processos de busca de solução;

(3) A solução deve ser encontrada dentro de um espaço de tempo aceitável.

A representação do problema por meio de um grafo, vai nos fornecer uma maneira simples de visualizar a forma como as técnicas de busca trabalham [Schildt,89]. A resolução do problema vai se constituir numa procura de um caminho através do grafo.

Um grafo representando o espaço de estados do problema, será constituído por nós que corresponderão a cada estágio de solução do problema e arcos que representarão as operações efetuadas para mudar de um estado para outro. Teremos dois estados, no grafo, que se destacarão particularmente: o estado inicial e o estado final do problema.

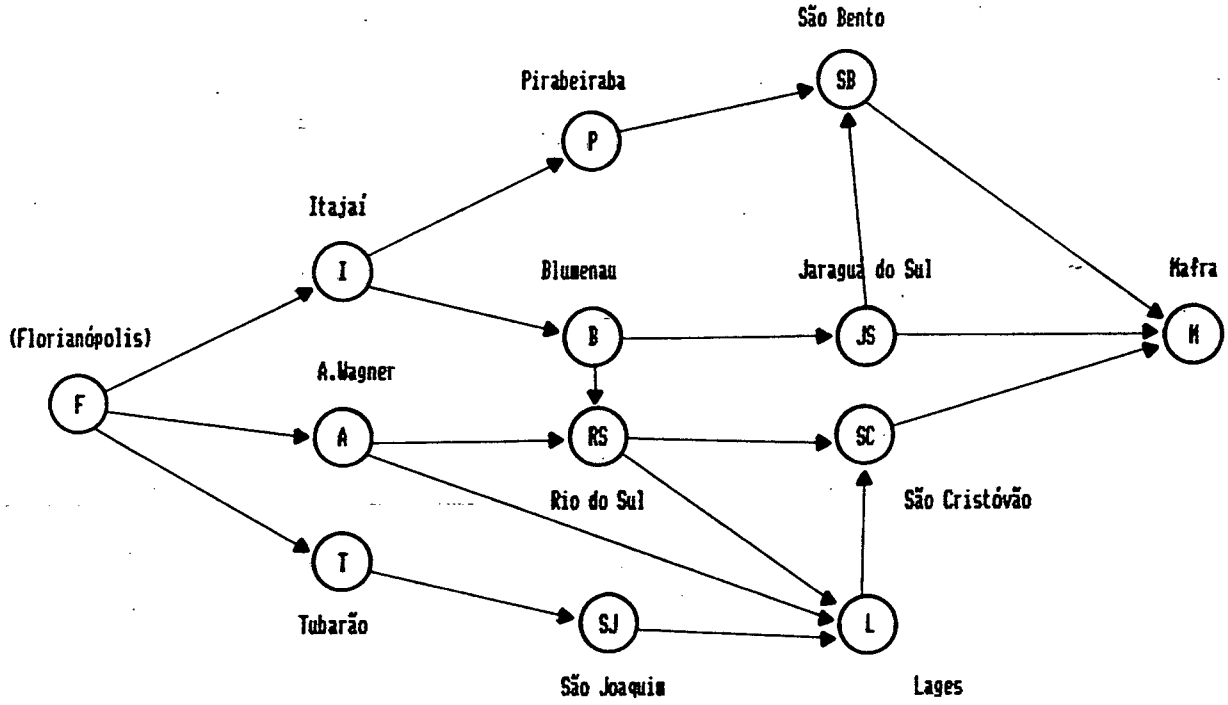
Seja, como exemplo a malha rodoviária dada na seguinte tabela:

INíCIO	FIM	DISTÂNCIA (Km)
Florianópolis	Itajaí	84
Florianópolis	Alfredo Wagner	101
Florianópolis	Tubarão	127
Itajaí	Pirabeiraba	105
Itajaí	Blumenau	47
Alfredo Wagner	Rio do Sul	78
Alfredo Wagner	Lages	115
Tubarão	São Joaquim	145
Pirabeirmba	São Bento	62
Blumenau	Jaraguá do Sul	50
Rio do Sul	São Cristóvão	93
Rio do Sul	Lages	126
São Joaquim	Lages	76
São Bento	Mafra	55
Jaraguá do Sul	São Bento	52
Lages	São Cristóvão	57
São Cristóvão	Mafra	189
Blumenau	Rio do Sul	98

Vamos resolver o seguinte problema: encontrar um caminho que leve de Florianópolis a Mafra.

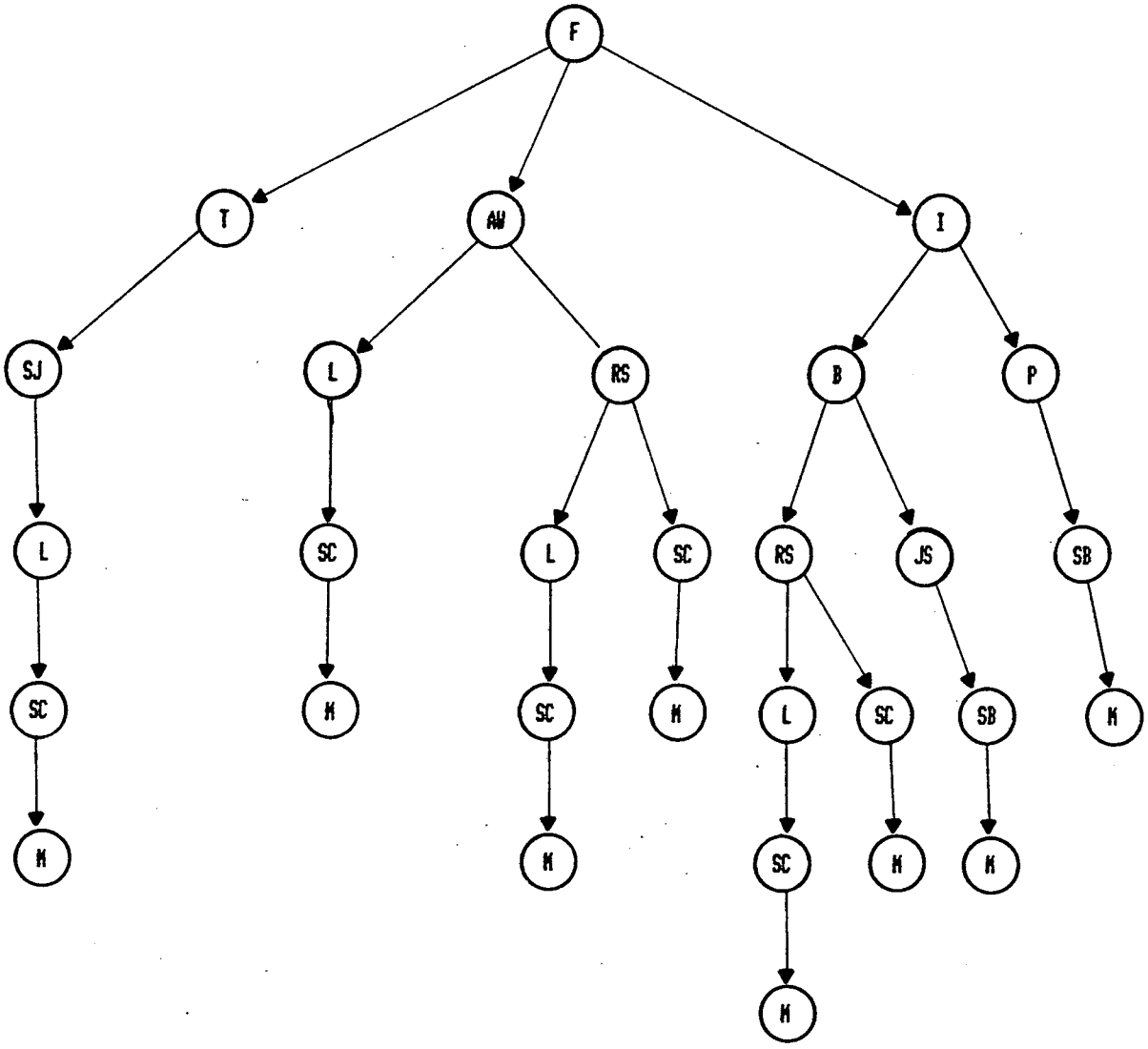
As cidades desta malha rodoviária são os estados do problema. O estado inicial é Florianópolis e o estado final é Mafra. Os operadores são as ligações entre cada cidade. Estes três elementos: o estado inicial, o estado final e os operadores, são os elementos que definem o espaço de estados.

A partir da tabela dada acima, podemos construir um grafo:



Partindo de Florianópolis, vários caminhos levam à Mafra. Mediante uma árvore (figura abaixo), teremos uma visualização destes caminhos. Uma árvore é um grafo orientado em

que cada nó tem somente um predecessor. O nó inicial é chamado de raiz.



Na árvore podemos identificar oito caminhos possíveis (para o problema dado), para ir de de Floríanópolis a Mafra. Para facilidade de entendimento outros traçados não foram acrescentados à malha viária, tampouco o foram os caminhos sem saída.

Um pequeno comentário deve ser feito neste ponto. Nem todas as partes do grafo serão determinadas durante o processo de resolução do problema. Conforme veremos, a busca da solução (ou o trajeto entre Florianópolis e Mafra), é feita concomitantemente enquanto se gera o grafo.

Alguns métodos para encontrar um caminho entre Florianópolis e Mafra serão vistos. Poderíamos, por exemplo, saindo de Florianópolis, tomar qualquer caminho, ir para outra cidade escolhida aleatóriamente, e, a partir dela, tomar qualquer outro caminho, até que por acaso chegássemos à Mafra. Este método poderá ser muito demorado, como tudo parece indicar. Será mais útil, portanto usar alguma estratégia apropriada. Tendo em vista que vamos usar um computador para resolver o problema, alguma estratégia inteligente de busca deve ser disponível para reduzir o espaço de busca, de forma a encontrar a solução correta para o problema, com um mínimo de computação.

Existem diferentes métodos que basicamente diferem pela ordem em que os nós são considerados. Vamos dividi-los em dois grupos:

(a) de busca cega;

(b) de busca heurística.

A diferença básica entre os dois grupos é o fato de que os métodos do segundo grupo usam o que chama "heurística" para reduzir o espaço do problema, minimizando ou maximizando algum componente ou aspecto deste problema.

3.4.1 Busca Cega

3.4.1.1 Busca em Profundidade

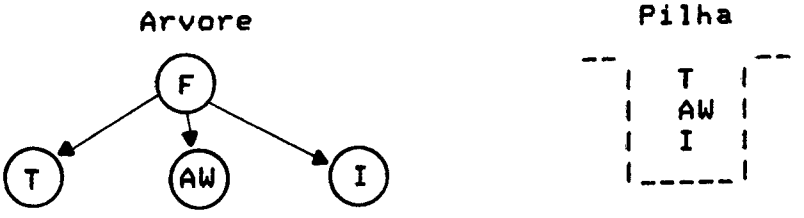
A busca em profundidade ou busca vertical explora cada caminho possível para chegar ao nó destino antes de tentar qualquer outro. Para entende-lo bem precisamos definir aqui uma estrutura chamada pilha, onde colocamos todos os nós gerados e retiramos sempre o de cima quando for necessário. Vamos usar ainda as definições de **nó aberto** e **nó fechado**. Convencionamos que o nó está aberto quando ele é gerado e colocado na pilha. Quando o nó for retirado da pilha ele é dito fechado e é colocado numa lista de nós fechados. Expandir um nó significa gerar seus sucessores, ou filhos. Por conseguinte, o nó antescendente a um determinado nó é dito pai deste nó.

Vamos iniciar a busca por Florianópolis, que é o nó raiz e representa o estado inicial do problema.

Ao mesmo tempo que vamos colocando os nós na pilha, vamos fazer uma representação de uma **lista_de_nós_abertos:**

lista_de_nós_abertos: [F]

Expandindo o nó F (Florianópolis), geramos: T (Tubarão), AW (Alfredo Wagner) e I (Itajaí), que vamos colocar na pilha, após retirarmos F (Florianópolis).



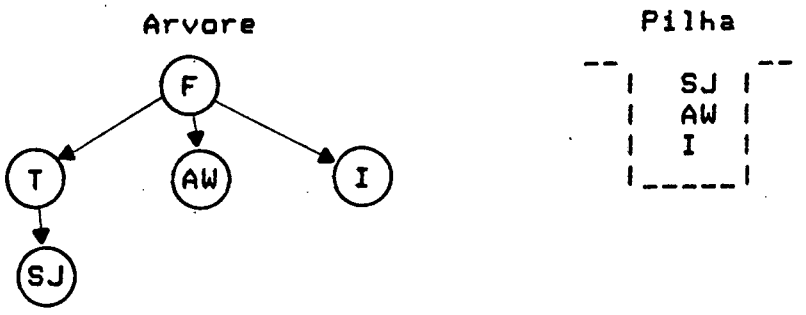
lista_de_nós_abertos: [T,AW,I]

Passamos agora, F (Florianópolis) para uma lista de nós fechados:

lista_de_nós_fechados: [F]

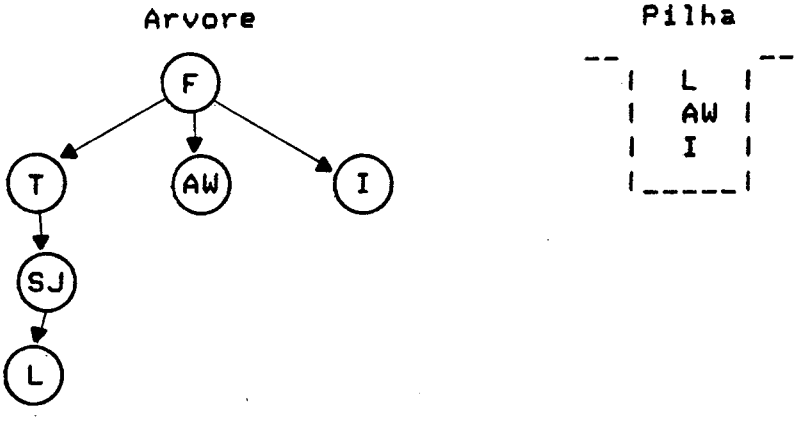
Examinamos agora o nó T (Tubarão) e geramos seus sucessores. Tiramos T da pilha e da **lista_de_nós_abertos** e o

colocamos na lista_de_nós_fechados:



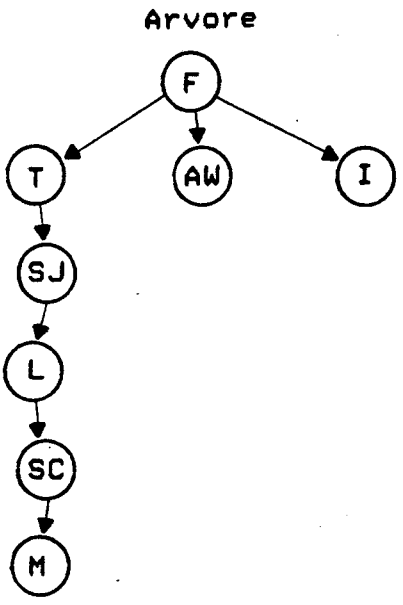
lista_de_nós_abertos: [SJ,AW,I]
lista_de_nós_fechados: [F,T]

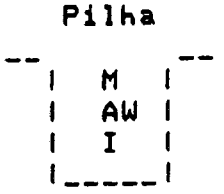
SJ será o próximo nó a ser expandido, pois é o que está no topo da pilha.



lista_de_nós_abertos: [L,AW,I]
lista_de_nós_fechados: [F,T,SJ]

Continuando este processo vamos chegar à seguinte configuração:





lista_de_nós_abertos: [M,AW,I]
lista_de_nós_fechados: [F,T,SJ,L,SC,M]

E chegamos a M (Mafra) que é o nó objetivo. O caminho percorrido foi:

Florianópolis-Tubarão-São Joaquim-Lages-São Cristóvão-Mafra

que pode-se inferir com facilidade, não é o menor caminho entre Florianópolis e Mafra, apesar de se constituir um trajeto turístico interessante.

A distância total percorrida neste trajeto será:

Florianópolis - Tubarão	:	127 Km
Tubarão - São Joaquim	:	145 km
São Joaquim - Lages	:	76 km
Lages - São Cristóvão	:	57 km
São Cristóvão - Mafra	:	189 km

Total	:	594 km

Convém em alguns casos de busca em profundidade, limitar o nível ou limite de profundidade de busca, para evitar uma pesquisa sem fim em determinado caminho ou um tempo muito grande de pesquisa que inviabilizaria todo o processo.

Barr e Feigenbam [Barr,81], descrevem o algoritmo de busca em profundidade com limite:

Passo 1: colocar o nó raiz numa lista de nós abertos. Se o nó for o nó objetivo, a solução foi encontrada.

Passo 2: se a lista de nós abertos estiver vazia, não existe solução para o problema.

Passo 3: tirar o primeiro nó n da lista de nós abertos e colocar numa lista de nós fechados.

Passo 4: se o limite de pesquisa foi encontrado, retornar ao passo 2.

Passo 5: expandir o nó n. Se ele não tiver sucessores, retornar ao Passo 2.

Passo 6: colocar todos os sucessores do nó n no início da lista aberta.

Passo 7: se algum dos sucessores do nó n é o nó objetivo, a solução foi encontrada. Senão, retornar ao passo 2.

3.4.1.2 Busca Horizontal

A busca horizontal, busca em largura, ou busca em nível, verifica cada nó de um mesmo nível antes de verificar nós de níveis seguintes. O Processo termina quando se acha o nó objetivo.

Vamos utilizar uma estrutura de fila, ao invés de pilha como no caso de busca em profundidade. A fila é uma estrutura de onde os nós serão retirados na mesma ordem em que entram, ou seja, o primeiro elemento que entra na fila, será o primeiro a sair.

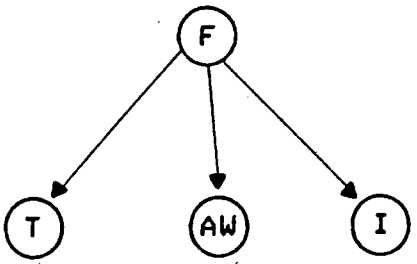
Para o problema que estamos resolvendo, o primeiro nó é F(Florianópolis):

Árvore

Lista_de_nós-abertos: [F] -----> fila

que vai gerar os seguintes sucessores: T (Tubarão), AW (Alfredo Wagner) e I (Itajaí).

Árvore



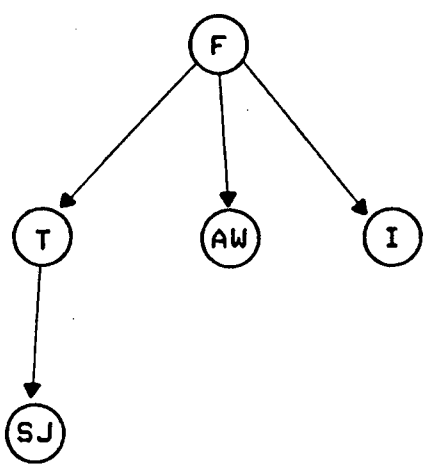
De forma similar, vamos construir duas filas: uma lista de nós abertos e uma lista de nós fechados:

Lista_de_nós-abertos: [T,AW,I]

Lista_de_nós_fechados: [F]

Tomamos agora o primeiro elemento da lista de nós abertos e o expandimos:

Árvore

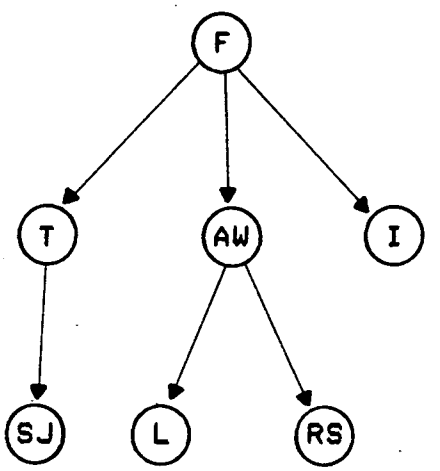


Lista_de_nós-abertos: [AW,I,SJ]

Lista_de_nós_fechados: [F,T]

O próximo nó a ser expandido é AW:

Árvore

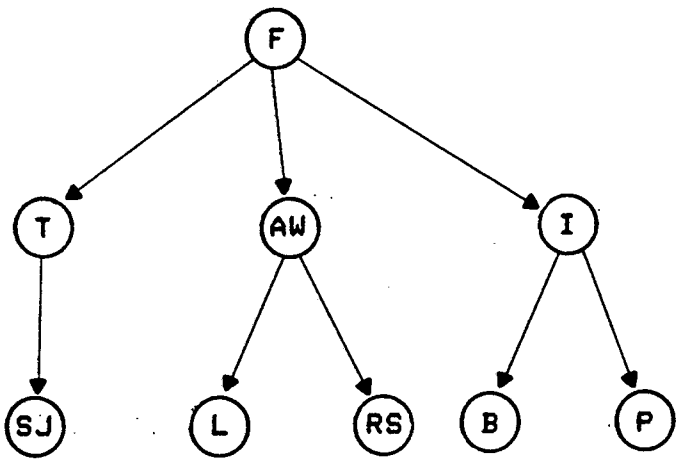


Lista_de_nós-abertos: [I,SJ,L,RS]

Lista_de_nós_fechados: [F,T,AW]

I (Itajaí) é o nó da frente da lista de nós abertos. Ele vai gerar os sucessores: B (Blumenau) e P (Pirabeiraba):

Árvore

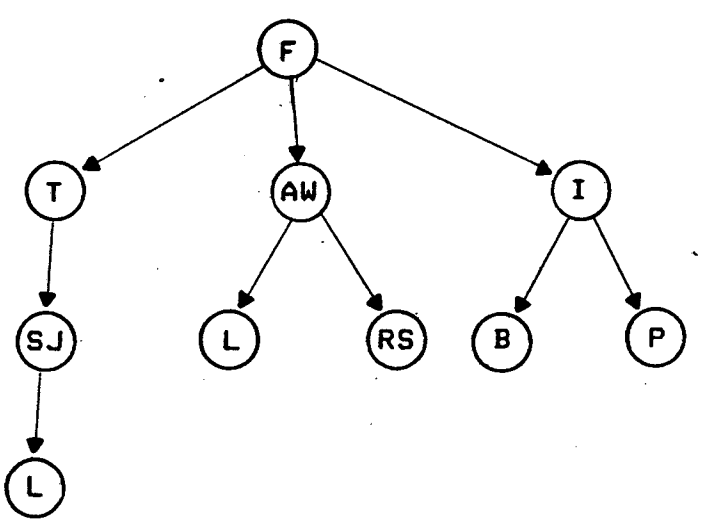


Lista_de_nós-abertos: [SJ,L,RS,B,P]

Lista_de_nós_fechados: [F,T,AW,I]

O nível 2, foi todo explorado. Passamos para o nível 3, gerando o sucessorer de SJ:

Árvore

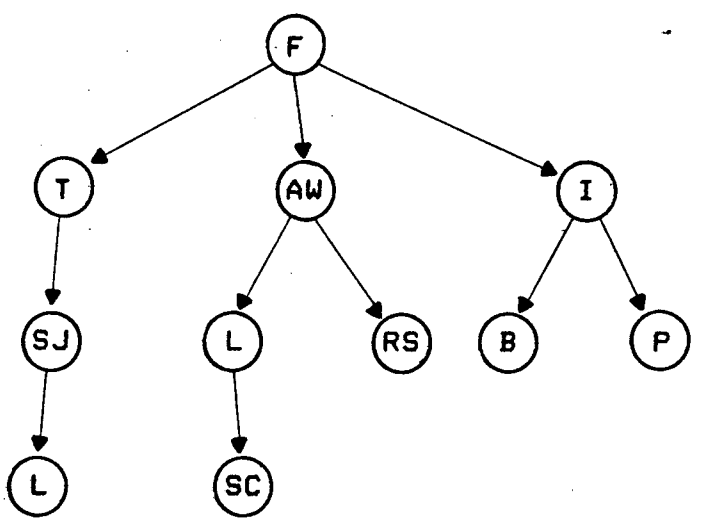


Lista_de_nós-abertos: [L,RS,B,P,L]

Lista_de_nós_fechados: [F,T,AW,I,SJ]

Tomamos o primeiro elemento da lista_de_nós_abertos L que gera SC (São Cristóvão). Retiramos L da lista_de_nós_abertos e o colocamos na lista_de_nós_fechados:

Árvore

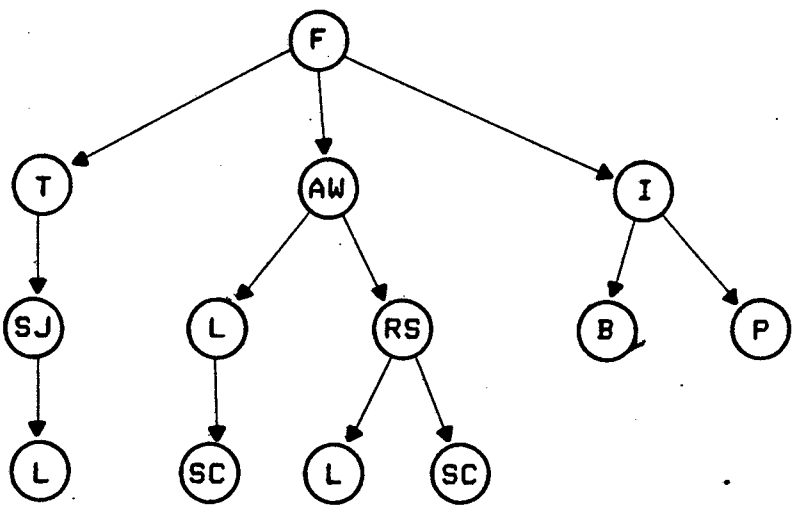


Lista_de_nós-abertos: [RS,B,P,L,SC]

Lista_de_nós_fechados: [F,T,AW,I,SJ,L]

O próximo nó a ser expandido é RS, que vai gerar: L (Lages) e SC (São Cristóvão):

Árvore

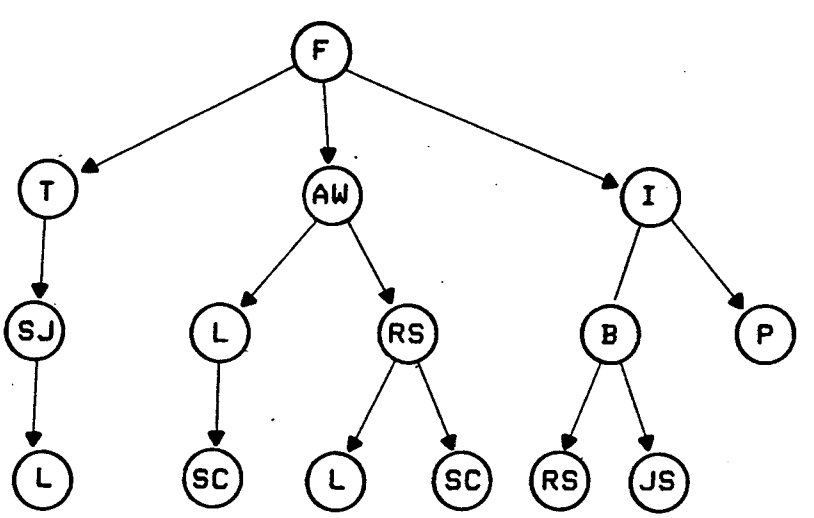


Lista_de_nós-abertos: [B,P,L,SC,L,SC]

Lista_de_nós_fechados: [F,T,AW,I,SJ,L,RS]

Próximo nó a ser expandido: B (Blumenau). Ele vai gerar os nós RS (Rio do Sul) e JS (Jaraguá do Sul):

Árvore

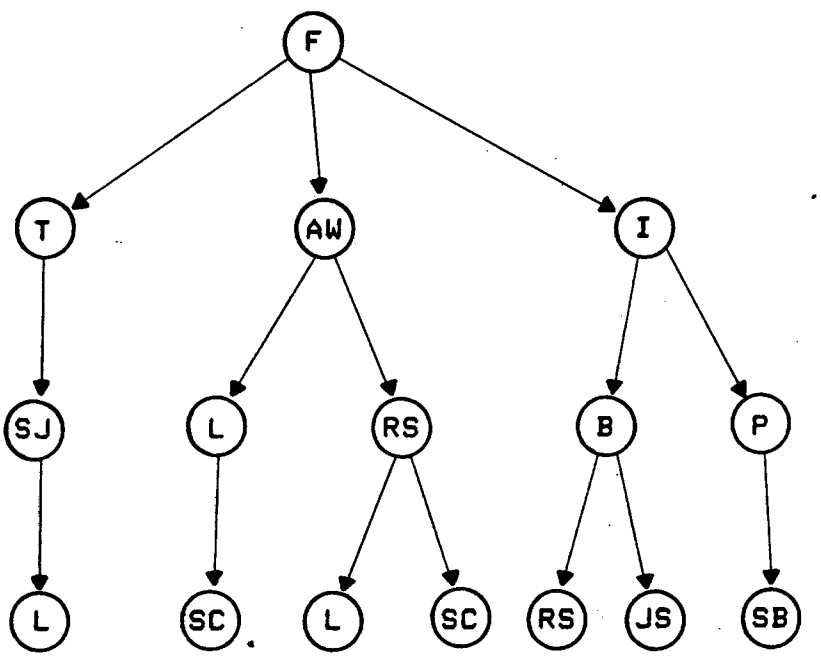


Lista_de_nós-abertos: [P,L,SC,L,SC,RS,JS]

Lista_de_nós_fechados: [F,T,AW,I,SJ,L,RS,B]

P (Pirabeiraba) está à frente da lista_de_nós_abertos. Ele vai gerar SB (São Bento).

Árvore



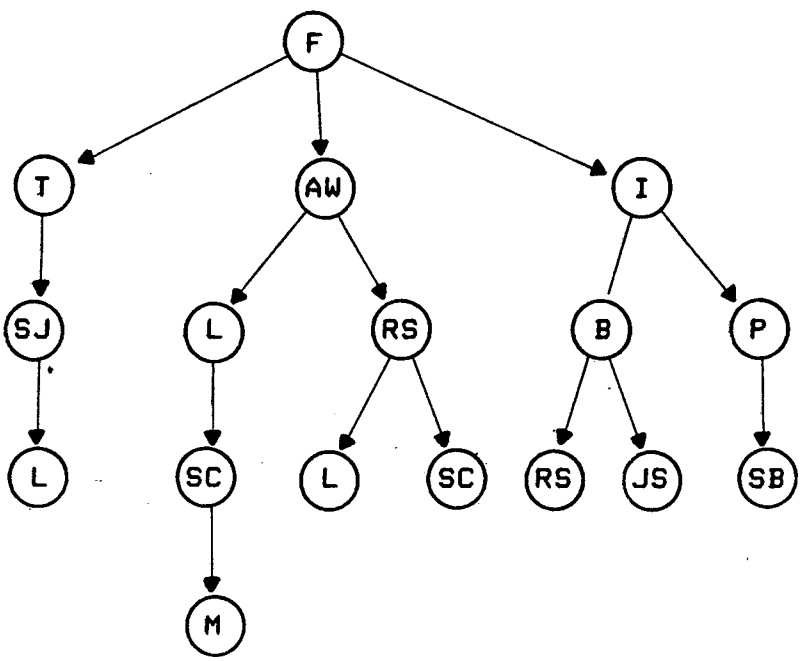
Lista_de_nós-abertos: [L,SC,L,SC,RS,JS,SB]

Lista_de_nós_fechados: [F,T,AW,I,SJ,L,RS,B,P]

Todos os nós do nível 2 geraram sucessores. Ainda não encontramos o nó objetivo. Passamos ao próximo nível. O primeiro elemento da lista_de_nós_abertos é L. Porém, L já havia sido "fechado", anteriormente. Eliminamos então L da lista_de_nós_abertos, passando ao nó seguinte, SC, que vai gerar M

(Mafra) que ó o nó objetivo.

Árvore



O caminho percorrido foi: F - AW - L - SC - M, ou seja, Florianópolis-Alfredo Wagner-Lages-São Cristóvão-Mafra. A distância total percorrida é:

Florianópolis - A. Wagner	:	101 km
A. Wagner - Lages	:	115 km
Lages - S. Cristóvão	:	57 km
S. Cristóvão - Mafra	:	189 km

Total		463 km

3.4.1.3 Avaliação dos Métodos de Busca Cega

Quando percorremos um grafo na busca de um caminho que leve ao objetivo, se pegar uma via qualquer for tão boa quanto outra, uma idéia cômoda é trabalhar com uma busca em profundidade [Winston,88]. Em situações que tivermos porém, ramos particularmente longos, a busca em profundidade poderá gastar muito tempo, podendo chegar a um beco sem saída. Neste caso seria interessante estabelecer níveis para pesquisa da busca.

A busca horizontal apresenta desvantagens quando tivermos casos em que o objetivo está muitos níveis abaixo do nó raiz [Schildt,89]. A busca em nível irá fazer um esforço considerável para encontrar este objetivo. No caso de todos os caminhos levarem ao nó destino, mais ou menos na mesma profundidade, a busca em largura é um desperdício [Winston,88].

3.4.2 Busca Heurística

O sucesso na resolução de problemas através de uma pesquisa em um grande labirinto de possibilidades inclui: a) a pesquisa seletiva neste labirinto de possibilidades e b) a redução das possibilidades a proporções manejáveis [Simon,81]. A busca cega pode levar a um tempo de processamento muito grande. Para evitar que isto ocorra, usamos a busca heurística. A palavra "heurística" é de origem grega e significa "servir para descobrir"

[Cunha,85].

Heurísticas são informações não necessariamente provadas, mas que podem conduzir a uma solução mais rápida do que uma busca exaustiva em todos os estados do problema. "Heurísticas são regras práticas aprendidas ou descobertas por especialistas de um dado domínio" [Carvalho,87]. Com uso de heurísticas, geralmente não está garantido o encontro de uma solução ótima para o problema, mas pode-se encontrar um resultado satisfatório em um tempo de processamento razoável. Algumas heurísticas podem ser boas para um problema específico e ruins para outros.

A heurística é, em geral, representada por números para poder permitir uma medida para se proceder as escolhas. Os valores a serem considerados para as heurísticas devem ser tais que a qualquer momento durante a execução do programa, sejam uma boa estimativa de avaliação para saber se se está no caminho certo. Em alguns casos a heurística procura minimizar valores e em outros procura maximizar [Rabuske,87].

As heurísticas podem surgir por pura intuição. é esta a forma como o homem resolve problemas quando não tem dados suficientes. é uma forma de conhecimento direto, em que a solução de um problema é encontrada repentinamente, não necessariamente de forma consciente, e às vezes, sem dados suficientes. Este "conhecimento" pode aparecer nas formas de suposição, pressentimento, antecipação ou adivinhação. Na história da

filosofia, das ciências e da técnica, muitas descobertas e invenções ocorreram por graças da intuição heurística [Bazarian,86].

3.4.2.1 Busca Subindo-Morro

O método de busca subindo-morro , subindo-a-colina ou subida-de-encosta, é em essencial a aplicação de uma heurística à busca em profundidade. Ao invés de colocar-se na pilha os sucessores de cada nó à medida que eles são gerados, usa-se uma heurística para ordenar na pilha estes sucessores, de forma que o nó no topo da pilha seja o nó que parece mais próximo do nó objetivo. O nome do método veio da analogia com o problema de um alpinista escalando uma montanha sob uma neblina muito espessa, com visibilidade praticamente nula. Nesta situação, o alpinista toma sempre um caminho mais íngreme, pois parece que este caminho é que o levará ao topo mais rapidamente.

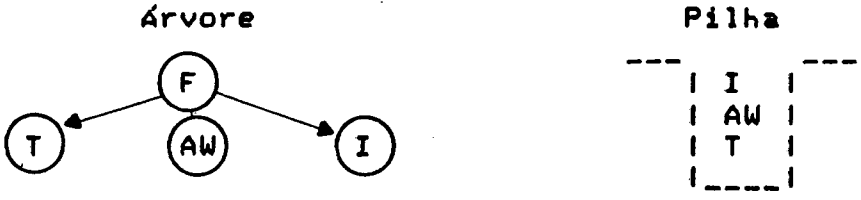
Para resolver o problema de encontrar um caminho entre Florianópolis e Mafra, inicia-se pelo nó raiz F(Florianópolis) e de forma análoga ao método de busca em profundidade, cria-se uma estrutura chamada pilha onde os nós serão colocados segundo as distâncias percorridas. As maiores distâncias irão no topo da pilha.

Para os sucessores de F(Florianópolis) , T e AW e I (Tubarão, Alfredo Wagner e Itajaí), coloca-se no topo da pilha, T (Tubarão), que está a 127 km de Florianópolis, distância maior do que se encontram Alfredo Wagner e Itajaí. 84 e 101 km, respectivamente. Escolhe-se o nó mais longe da origem, na esperança de que ele esteja mais próximo do destino que os demais.

3.4.2.2 Busca de Menor Custo

O método de busca de menor custo ou descendo-a-ladeira, usa uma heurística oposta à heurística do método subindo morro. Ao invés de ordenar-se colocando no topo da pilha o nó que está mais longe do nó raiz, coloca-se o nó mais próximo do nó raiz buscando-se minimizar a distância percorrida, ou seja, procurando-se o caminho de menor esforço.

Os sucessores de F são: T, AW e I.

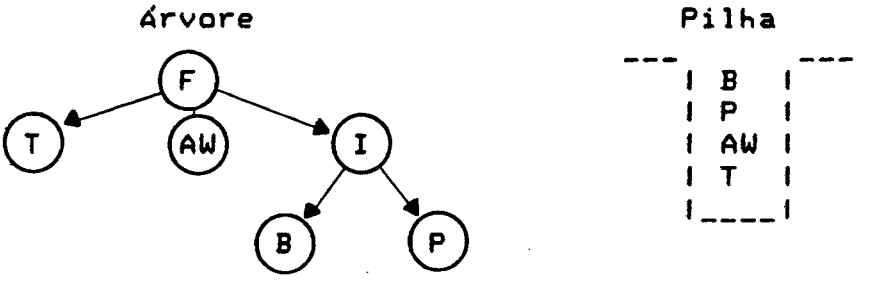


INÍCIO	FIM	DISTÂNCIA (Km)
Florianópolis	Itajaí	84
"	A. Wagner	101
"	Tubarão	127

Lista-de-nós-fechados: [F]

Ordena-se a pilha, em ordem crescente, de distâncias percorridas.

O nó a ser expandido é I, que vai gerar: B (Blumenau) e P (Pirabeiraba).

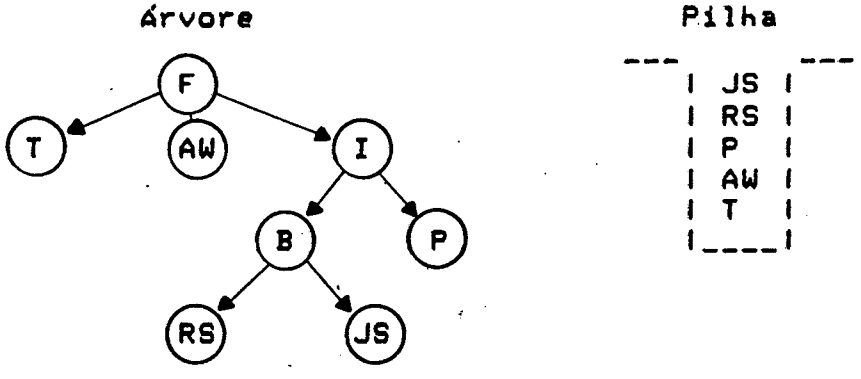


Lista-de-nós-fechados: [F,I]

As distâncias percorridas são:

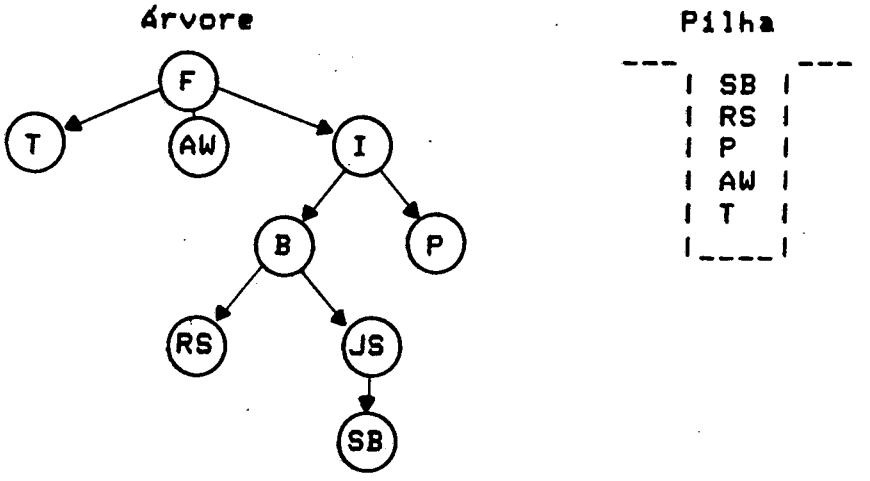
INÍCIO	FIM	DISTÂNCIA (km)
Itajaí	Pirabeiraba	105
"	Blumenau	47

Como Blumenau está a 47 km (Pirabeiraba está a 105 km), fica no topo da pilha. Continuando o processo de busca, obtemos: RS (Rio do Sul) e JS (Jaraguá do Sul) a 87 e 50 km, respectivamente. Logo, obtém-se:



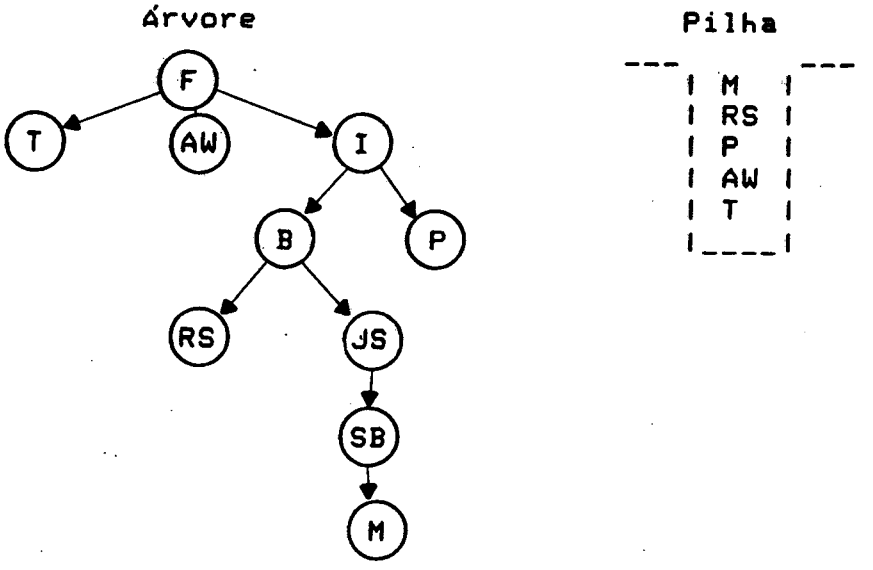
Lista-de-nós-fechados: [F,I,B]

Expandindo o nó JS (Jaraguá do Sul):



Lista-de-nós-fechados: [F,I,B,JS,SB]

JS vai gerar SB, e SB vai gerar M (Mafra) que é o nó objetivo.



Lista-de-nós-fechados: [F,I,B,JS,SB]

O caminho percorrido é:

Florianópolis - Itajaí - Blumenau - Jaraguá do Sul - São Bento - Mafra.

A distância total percorrida é:

Florianópolis - Itajaí	:	84 km
Itajaí - Blumenau	:	47 km
Blumenau - Jaraguá do sul	:	50 km
Jaraguá do Sul - São Bento	:	52 km
São Bento - Mafra	:	55 km

Total	:	288 km

3.4.2.3 Avaliação dos Métodos de Busca Heurística

O método de busca "subindo o morro" procura reduzir o número de nós que devem ser explorados até encontrar uma solução. Este método porém apresenta alguns problemas:

a) o problema dos contrafortes;

b) o problema dos platôs;

c) o problema das serras.

O problema dos contrafortes ou dos falsos morros ocorrem quando existem "picos" secundários que podem levar a uma falsa conclusão.

O problema dos platôs ocorre quando, fazendo-se uma analogia com o relevo, aparece uma grande zona plana entre os picos, ou seja, todos os próximos passos parecem igualmente bons

[Winston,88] e [Schildt,89]. Não há indicativas de qual a melhor direção de deslocamento.

O problema das serras ou das cristas ocorre quando existe no espaço de busca uma área mais alta que as áreas circundantes, que não pode ser atravessada com facilidade.

Problemas que ocorrem na busca de menor custo são o inverso dos de subindo o morro, ou seja, vamos ter vales, planícies e gargantas.

3.5 Instrumentos de Inteligência Artificial

Os instrumentos de IA podem ser classificados de uma forma geral em linguagens e ambientes. As linguagens recebem de alguns autores a classificação de linguagens de programação gerais e linguagens de programação específicas, e dentro delas Lisp se incluiria na primeira classificação e Prolog na segunda. Outros autores já acham que Prolog ficaria entre linguagem e ambiente, haja visto que dispõe de algumas características de ambiente tais como máquina de inferência e um método de representação do conhecimento.

Os ambientes são utilizados na construção de sistemas especialistas. São mais ou menos rígidos. A interface com o usuário é fornecida pelo sistema que só permite a construção da base de conhecimento. Alguns ambientes apresentam interfaces sofisticadas e permitem a formalização do conhecimento de formas mais variadas, e às vezes são denominadas de ambientes de programação, quando tem estas características, para distinguí-los dos demais. Os ambientes sem estas características são chamados de "Shells". Independente porém da sua sofisticação, uma shell é um sistema onde falta acrescentar a base de conhecimento. Dispõe de uma máquina de inferência e de interfaces, tais como o módulo explicativo. Sobre os instrumentos ver [Harmon,88] e [Waterman,86].

Um programa de IA pode ser construído em qualquer linguagem de programação, mas algumas linguagens tem características que as tornam próprias para este fim, como é o caso de Lisp e Prolog [Rich,88]. Estas características são:

a) facilidade de manipulação de listas;
b) manipulação de estruturas cujo tamanho e tipo não se sabe a priori;
c) capacidade de armazenamento dinâmico de dados para serem usados em dedução;
d) boas estruturas de controle.

3.5.1 Lisp

Lisp (List Processing), atualmente a mais popular linguagem de Inteligência Artificial, foi criada em 1958 por John McCarthy do MIT. Foi desenvolvida a partir do IPL - Information Processing Language. O IPL, desenvolvido por J. C. Shaw, Allen Newell e Herbert Simon, tinha sido a primeira linguagem de computação com o objetivo de permitir o processamento simbólico.

Existem praticamente versões de Lisp para quase todos os tipos de computadores. Mas, apesar da existência de tantos dialetos, existe uma tendência de se caminhar para uma padronização de Lisp através do COMMON LISP.

Em Lisp há umas existem poucas funções básicas, escritas em linguagem de máquina. Grande parte das outras funções são escritas usando-se estas funções básicas. Um usuário acostumado com determinadas funções num dialeto de Lisp, se mudar para outro dialeto pode recriar estas funções. As funções básicas, escritas em linguagem de máquina, são praticamente comuns em todos os dialetos.

As funções Lisp são pré-fixas, ou seja, as funções precedem os seus argumentos. Assim, a função que obtem a soma de dois ou mais argumentos é dada por:

(PLUS ⟨arg1⟩ ⟨arg2⟩ ...) ou (+ ⟨arg1⟩ ⟨arg2⟩...).

Lisp permite formas de inibir a avaliação de funções bem como provocar a avaliação de funções e argumentos. Possui algumas funções especiais denominadas predicados cuja avaliação pode ser verdadeira (true) ou falso (nil).

As principais características de Lisp, são:

a) processamento simbólico com estruturas de tamanho ilimitado;
b) a estrutura básica é a lista. Não tem declarações de variáveis;
c) tem sempre a mesma representação. Programa é função em Lisp. Programas em Lisp podem usar

outros programas como dados.

d) a linguagem é recursiva;

e) permite boa interatividade com o usuário.

Lisp apresenta algumas desvantagens:

a) é dificil construir em Lisp expressões matemáticas complexas;

b) necessita de muitos comentários para identificar as funções, caso contrário, quando voltar a elas, o próprio programador terá dificuldades de entender o que foi feito;

c) usa muita memória.

3.5.2 Prolog

Prolog (Programing in Logic) foi criada em 1972 por A. Colmerauer e P. Roussel, na Universidade de Marselha, França. É uma linguagem capaz de fazer deduções a partir do cálculo dos predicados. Tal como LISP, Prolog destina-se à computação simbólica.

Prolog é próprio para implementação de sistemas especialistas do tipo sistema de produção. Neste tipo de sistema, as regras são do tipo SE-ENTÃO:

SE

⟨premissas⟩

ENTÃO

⟨conclusão⟩

As premissas podem ser chamadas também por lado esquerdo, antecedentes ou corpo da regra. A conclusão pode ser chamada de consequente, lado direito ou cabeça da regra.

Prolog não tem comandos de atribuição de variáveis. Elas recebem valores através do que se chama de instanciação. Quando uma variável é instanciada, ela passa a representar o mesmo valor em toda uma mesma regra, e apenas nela. Noutra regra, variáveis de mesmo nome estarão livres, podendo instanciar outros valores, que dependem do processamento do programa.

3.6 Aplicações de Inteligência Artificial

O campo da Inteligência Artificial abrange vários sub campos de pesquisa:

- processamento de linguagem natural;
- visão computacional;
- jogos;
- sistemas especialistas;
- robótica.
- redes neurais

3.6.1 Processamento de linguagem natural

A habilidade de comunicação dos seres humanos é demonstrada de várias maneiras: linguagens, expressões faciais e corporais, gestos etc. A linguagem é entretanto a forma mais poderosa de comunicação dos seres humanos e é uma das mais importantes diferenças entre estes e os animais inferiores." O homem tem várias vantagens sobre os animais, como, por exemplo, o fogo, o vestuário, a agricultura e as ferramentas... Mais importante, porém, do que qualquer uma dessas coisas, é a linguagem" [Bertrand Russel, Deliamentos da Filosofia, Civilização Brasileira, 3a. edição, 1969].

Processamento de linguagem Natural (PLN) pode ser definido como tentar fazer o computador entender comandos escritos numa linguagem natural tais como: ingles, chines ou portugues. A função de um processador de linguagem natural é extrair informação da sentença de entrada, independente da forma como esta sentença entra no computador [Schildt,89].

O processamento de linguagem natural é uma das grandes áreas de pesquisa em Inteligência Artificial. Os cientistas de informática juntaram-se a linguistas para criar programas que entendessem a linguagem humana. Os primeiros programas eram tradutores de linguas, mas não apresentaram muito bons resultados no início, haja visto que a própria linguística estava nascendo.

A linguística praticamente começou em 1958 com o livro "Syntactic Structures", de Noam Chomsky. Com o desenvolvimento de gramáticas estruturais da linguagem humana, a linguagem natural pode ser descrita como um conjunto de regras sintáticas e estruturais. Estas regras dividem as frases em componentes que são utilizáveis pelos programas [Hilster,junho/87].

"A capacidade dos computadores em entender e interpretar a linguagem natural e humana torna-os mais amigáveis, porque permite aos usuários se comunicarem com os programas de computação usando a sua própria língua nativa" [Hilster,nov/87,pág.66].

O uso de linguagem natural torna mais fácil e mais acessível o uso de computadores pelos usuários. Futuramente, o homem falará com o computador e vice_versa [Siqueira,87].

Uma interface de linguagem natural tem a vantagem de permitir que se faça a mesma pergunta de várias formas. Isto aumenta as chances de se ter a informação que se quer e em caso contrário, pode-se ainda formular a pergunta de forma diferente.

Um dos aspectos da linguagem natural que a torna mais atraente, é a possibilidade de se construir módulos no programa que permitam identificar palavras que não estão no banco de dados, mas que podem ser introduzidas pelo usuário . Deste modo, o usuário pode criar a sua própria biblioteca de vocábulos que tenha uso mais frequente.

3.6.2 Jogos

A noção de que os computadores poderiam competir em jogos existe há tanto tempo quanto os computadores. Shannon, em 1950, escreveu na Philosophical Magazine, o artigo "Programming a Computer for Playing Chess", onde descrevia mecanismos que poderiam ser utilizados num programa para jogar xadrez. No xadrez, para heurística, poderia ser levado em consideração o valor das peças, sua mobilidade e posição. Shannon estabeleceu a base do que viria mais tarde a constituir a espinha dorsal de muitos outros

programas [Santos,85]. No início dos anos 60, Arthur Samuel escreveu um programa que jogava damas e que aprendia com seus próprios erros.

Para se jogar bem xadrez, deve-se ter a capacidade de visualizar jogadas subsequentes a um determinado movimento. O ser humano quanto muito poderia visualizar talvez até sete movimentos depois do movimento dado. Quem sabe, alguém poderia visualizar mais. De qualquer forma, muitas alternativas de movimentos subsequentes não seria possível de serem consideradas. Os jogadores de xadrez, escolhem então as que parecem produzir os melhores resultados. Os computadores também podem proceder desta forma, com a vantagem de poder verificar um número muito grande de alternativas possíveis. E computadores que jogam xadrez tem conseguido vencer nos jogos , as próprias pessoas que os programaram [Ullrich,87]. é o efeito da magnitude dos problemas.

Porém, o objetivo principal desses programas de jogos, não é demonstrar que as máquinas poderiam ser inteligentes e vencer humanos como seus adversários, e sim extrair das pesquisas, teorias gerais sobre a inteligência e como implementá-la. A partir dos trabalhos executados nesta área, principalmente em jogos de xadrez, surgiram muitas das técnicas utilizadas hoje, na construção dos sistemas especialistas. Os princípios e técnicas apreendidos destes trabalhos, permitiram aos pesquisadores explorar outras áreas tais como medicina, química, geologia e biologia.

3.6.3 Visão Computacional

A visão computacional abrange a compreensão e interpretação de cenários a partir de imagens, pelos computadores.

A visão computacional começou a ser estudada nos anos 60. Os primeiros trabalhos envolviam reconhecimento de letras. Em 1965, L. G. Roberts, no Massachusets Institute of Technology, construiu um programa que processava imagens tridimensionais de poliedros. O programa localizava beiras, linhas e cantos. A maior parte das técnicas usadas desde então, é resultado daquele trabalho.

3.6.4 Sistemas Especialistas

Os sistemas especialistas começaram a surgir comercialmente no início da década de 80. A primeira empresa que foi formada com o objetivo exclusivo de construir sistemas especialistas foi a Intell Genetics, com técnicos da Universidade de Stanford, para o campo de engenharia genética.

Sistemas especialistas são programas com técnicas de IA que capacitam um computador a resolver problemas tal como um perito humano, sobre determinado tema que não faz parte do conhecimento comum. Peritos humanos são pessoas extremamente competentes na solução de tipos específicos de

problemas [Weiss,88].

O knowhow do perito humano é utilizado para instruir o computador a resolver um problema ou a tomar uma decisão. Este knowhow pode ser transformado em regras adicionadas ao programa. É claro que algumas características inerentes ao ser humano, tais como experiência, intuição e principalmente bom senso, dificilmente poderão ser transportados para o computador [Lorenzoni,nov/88].

Segundo Williams [Williams,87], os atributos desejáveis para um sistema especialista são os seguintes:

a) um sistema especialista deve conter separados entre si o conhecimento sobre um domínio específico e a metodologia de resolução de problemas;

b) um sistema especialista deve "pensar" da mesma forma que um ser humano;

c) um sistema especialista deve aprender com a experiência;

d) um sistema especialista deve ter uma base de conhecimento modificável com possibilidade de ser dividida em módulos diferentes;

e) um sistema especialista deve interagir com o usuário em linguagem natural;

f) um sistema especialista deve ter uma estratégia de controle simples e transparente para o usuário;

g) um sistema especialista deve ser capaz de resolver problemas rapidamente e não exigir recursos de hardware muito caros;

h) um sistema especialista deve ter dispositivos para ajudar o diálogo com o usuário;

i) um sistema especialista deve ter uma interface orientada para monitores;

j) um sistema especialista deve ser capaz de raciocinar sobre incerteza e insuficiência de dados sobre o problema.

Segundo Waterman [Waterman,86], os sistemas especialistas, de uma forma geral, se aplicam em: a) previsão; b) diagnóstico; c) projeto; d) planejamento; e) instrução; f) controle. A maioria dos sistemas especialistas construídos nos últimos dez anos foram voltados para a área médica, análise química, exploração geológica, solução de configuração de computadores e diagnósticos de falhas de computadores.

Um sistema especialista é dividido basicamente nas seguintes partes:

- um banco de dados que contém fatos e regras sobre um determinado domínio;
- um mecanismo de inferência que é a parte do programa que utiliza o banco de dados procurando uma ou mais soluções para o problema;
- uma base de dados dinâmica que contém os dados de entrada e se atualiza mantendo os dados sobre o estado atual da solução do problema;
- uma interface para manter diálogo com o usuário para entrada de dados e saída de resultados;
- um módulo explicativo sobre a forma de se chegar a uma conclusão.

Os sistemas especialistas podem oferecer conhecimento onde os especialistas humanos não são possíveis [Hilster,nov/87]. Além disso, os especialistas humanos não são perenes. Podem se transferir para outras empresa e se aposentarem. Os sistemas especialistas são facilmente transferidos, reproduzidos e são relativamente permanentes.

3.6.5 Robótica

A palavra robô que deriva do substantivo slavo "robota", significa trabalho em russo e faxina em tcheco. Foi usada pela primeira vez, em 1923, na peça de teatro "Rossum's Universal Robots", escrita pelo dramaturgo tcheco Karel Capeck. Nesta peça, o autor satiriza a civilização mecanizada da época, e descreve máquinas andróides fabricadas por Rossum, um brilhante cientista para realizar tarefas tediosas para a humanidade [Araribóia,88]. Após terem sido usados na guerra, um colega de Rossum, dota os robôs de emoções e estes se revoltam contra os homens por serem tratados como escravos e dizimam a humanidade. A partir daí, este nome foi-se vulgarizando pela ficção científica, passando a designar humanóides ou máquinas com formato que possa lembrar o corpo humano, dotadas de capacidade de deslocamento, ação e decisão, superinteligentes, lançadores de raios, etc. Alguns autores trataram os robôs como máquinas hostis aos homens. Outros, ao contrário, procuraram mostrar que os robôs, não tinham que ser necessariamente as máquinas hostis, sugeridas por Capek, como foi o caso de Isaac Asimov. Asimov procurou retratar robôs como máquinas sensíveis, com as quais o homem poderia se envolver emocionalmente. A ele se deve o termo "robótica", para designar o estudo e a construção de robôs.

O Instituto de Robôs da América (Robot Institute of America) define robô como um equipamento multifuncional e reprogramável, projetado para movimentar materiais, peças,

ferramentas ou dispositivos especializados através de movimentos variáveis e programados, para a execução de muitas tarefas. [Ullrich,87].

Os robôs podem ser classificados de acordo com o seu sistema de controle. Groover [Groover,89], dá a seguinte classificação: a) robôs de sequência fixa; b) robôs de repetição com controle ponto a ponto; c) robôs de repetição com controle de trajetória contínua; d) robôs inteligentes. Os robôs inteligentes, constituem a classe de robôs que tem a capacidade de interação com o meio-ambiente, ou seja, ele dá respostas na forma de movimentos não previstos com antecedência às informações provenientes do ambiente de trabalho, relativas a obstáculos e contornar, distâncias de peças e/ou obstáculos, etc. Seu controle é executado por computadores. Eles podem assim, realizar uma série de funções que não são especificadas previamente pelo projetista, diferentemente da automação convencional que realiza apenas uma única função. Eles podem ser programados para determinadas tarefas e podem ser reprogramados para outras.

Os robôs inteligentes podem ser equipados para sentir calor, pressão, etc, via sensores, bem como podem ser dotados de sistemas de visão. Sensores são dispositivos que associam grandezas de fácil tratamento como os sinais elétricos à grandezas não utilizáveis diretamente pelo computador tais como velocidade, aceleração, distância dos objetos, condições ambientais, etc [Ferreira,87].

Dois tipos de sensores são utilizados: sensores internos e sensores externos. Os sensores internos determinam os ângulos formados pelas juntas das articulações dos robôs. Os sensores externos são classificados em sensores de contato e sensores de não-contato. Uma das funções dos sensores de contato é o tato. Os sensores de não-contato tem o objetivo de determinar distâncias, formas externas de objetos, etc. O principal sensor de não-contato é a visão [Araribóia,88].

Nós vemos os objetos não por causa deles, e sim pela luz que eles refletem. A luz que passa pela lente dos olhos é focalizada na retina. A retina é composta de minúsculos receptores que convertem a luz em sinais elétricos.

Sensores de visão estão num estágio primário de desenvolvimento mas graças ao volume de investimentos feitos na área, seu desenvolvimento está se acelerando. O sistema de visão é o elemento chave para a obtenção de robôs mais versáteis [Ferreira,87].

Para equipar os robôs com os sentidos táteis, podem ser utilizados sensores de pressão, para que estes sintam a força ao segurar. Dispositivos sensíveis ao calor podem ser usados para diferenciar os objetos quentes dos objetos frios.

Uma grande possibilidade de utilização de robôs, se dotados de grande número de funções motoras, inteligência e

energia suficiente para uma autonomia adequada, seria seu uso em trabalhos em ambientes hostis ou perigosos ao homem. Sua invulnerabilidade à doenças, calor, radiação, etc, tornam o seu uso de uma importância estratégica. Eles poderiam como auxiliares em laboratórios lidar com bactérias e virús, evitando-se assim o risco desnecessário de submeter seres humanos à infecções e doenças contagiosas. Alguns setores da indústria, mormente aquelas ligadas à fundição onde as temperaturas ambientais são altíssimas e os ruídos ultrapassam os níveis estabelecidos pela medicina como suportáveis sem causar danos ao homem, estão dentre as possibilidades de uso dos robôs.

Como mencionado anteriormente, a invulnerabilidade dos robôs à radiação, permitirá seu uso em inspeção de reatores nucleares. Onde haja risco de vida, seus serviços serão indispensáveis.

A pesquisa espacial será impulsionada pelo desenvolvimento da robótica. Robôs serão usados na mineração de asteróides e de outros planetas. No momento procura-se desenvolver robôs para reparar satélites em órbita [Ullrich,87]. Em asteróides e planetas com menor gravidade que na terra, os robôs poderão movimentar grandes cargas e tem no sol um suprimento de energia inesgotável. É claro que as vantagens do seu uso no espaço, em ambientes hostis ou pelo menos desconfortáveis ao homem são muitas. Mas, os homens serão indispensáveis no espaço. A habilidade de improviso é uma capacidade inerente aos seres

humanos e ela não é transferível para os robôs.

Na área militar, nos Estados Unidos, a robótica tem sido desenvolvida com vários objetivos. Entre eles: criar uma sentinela capaz de vigiar uma grande área com perfeita visão noturna e construir veículos robotizados que possam transpor obstáculos, sentindo e respondendo ao inesperado, com memória dinâmica, onde os dados são atualizados continuamente de acordo com as suas próprias experiências. O objetivo principal destes veículos não tripulados, será seguir outros veículos em terrenos acidentados [MIKRO,fev/89].

Na década de 90 (a previsão é para 1994) a União Soviética pretende enviar robôs guiados por IA para Marte.

3.6.6 Redes Neurais

Na antiguidade, filósofos gregos como Platão e Aristóteles já tentavam explicar o funcionamento do cérebro humano. No século XVIII, Descartes fez investigações sobre os processos mentais. Estes estudos são interessantes do ponto de vista histórico. Eles estiveram longe de produzir quaisquer resultados práticos.

Por volta dos anos 40, W.S. McCulloch e W.A. Pittis começam a estudar a computação neural, ou seja, a simulação em computador do funcionamento do cérebro humano.

O neurônio é o principal componente do cérebro. Do seu corpo sai um filamento alongado chamado axônio, através do qual cada célula se liga às outras. E, estas ligações são feitas por intermédio de pequenos filamentos denominados dentritos. A estas ligações dos dentritos com os neurônios chamamos sinapses. Milhões de neurônios ligados por sinapses formam uma rede neural.

Os neurônios podem se encontrar em dois estados diferentes:

(a) ativos (ou excitados);
(b) inativos (ou inibidos).

Um neurônio está ativo, quando ele envia um sinal elétrico excitatório através dos axônios para outros neurônios. Caso contrário ele estará inibido (ele envia um sinal elétrico inibitório).

Poderíamos representar um agrupamento de neurônios por uma lista de números que representassem os estados destes neurônios, que poderiam estar excitados ou inibidos. Poderíamos convencionar, por exemplo:

inibido --> -1
excitado --> 1.

Rodovias poderiam ser representadas por:

rodovias --> neurônios R1, R2, R3, R4, R5.

A BR-101 poderia ter a seguinte configuração:

BR-101 --> E1 --> inibido
E2 --> inibido
E3 --> excitado
E4 --> excitado
E5 --> excitado

Com a convenção adotada, teríamos assim a seguinte lista, representando a BR-101:

[-1,-1,1,1,1]

Memórias Associativas

A fim de ter-se uma idéia mais clara do funcionamento dos neurônios artificiais, vamos construir um sistema rudimentar de rede neural para memória associativa, ou seja, associar conceitos.

Sejam as duas listas abaixo:

(a) BR-101, SC-404, FLN-010;

(b) rodovia federal , rodovia estadual , rodovia municipal.

onde em (a) temos uma lista de rodovias e em (b) a condição destas rodovias quanto à sua jurisdição, ou seja: se a rodovia é federal, estadual ou municipal.

Para representar rodovias e jurisdição, vamos utilizar dois grupos diferentes de neurônios: sendo um grupo com cinco neurônios e o outro com quatro neurônios.

rodovias -----> neurônios R1, R2, R3, R4, R5

jurisdição -----> neurônios J1, J2, J3, J4

Vamos convencionar como representação destes conceitos as listas abaixo:

(a) BR-101 : [1,1,-1,-1,1]
SC-404 : [-1,-1,1,1,1]
Fln-010: [-1,1,-1,1,-1]

(b) rodovia federal : [1,1,-1,-1]
rodovia estadual : [1,-1,1,-1]
rodovia municipal: [-1,1,1,-1]

Os pares de associação a serem estabelecidos são:

BR-101 -----> rodovia federal

SC-404 -----> rodovia estadual

FLN-010 -----> rodovia municipal

Para cada par de associação, vamos construir o que se chama tabelas de reforço, com a seguinte configuração:

	R_1	R_2	R_3	R_4	R_5	
J_1	[[	,	,	,	,	],
J_2	[	,	,	,	,	],
J_3	[	,	,	,	,	],
J_4	[	,	,	,	,	]].

Esta tabela de reforço (uma matriz), pode ser representada como uma lista de listas em Prolog. A construção desta tabela, deve seguir os seguintes critérios:

(a) se R_i e J_i estiverem ambos excitados ou ambos inibidos, colocamos 1 no cruzamento da linha J_i com a coluna A_i.

(b) caso contrário, colocamos -1.

Tabela de reforço para o par BR-101 - rodovia federal

BR-101 : [-1,-1,1,1,1]

rodovia federal : [1,1,-1,-1]

	R_1	R_2	R_3	R_4	R_5	
J_1	[[-1	-1	1	1	1	],
J_2	[-1	-1	1	1	1	],
J_3	[1	1	-1	-1	-1	],
J_4	[1	1	-1	-1	-1	]].

Tabela de reforço para o par SC-404 - rodovia estadual

SC-404 : [1,1,-1,-1,1]

rodovia estadual : [1,-1,1,-1]

	R_1	R_2	R_3	R_4	R_5	
J_1	[[1	1	-1	-1	1	],
J_2	[-1	-1	1	1	-1	],
J_3	[1	1	-1	-1	1	],
J_4	[-1	-1	1	1	-1	]].

Tabela de reforço para o par FLN-010 - rodovia municipal

FLN-010 : [-1,1,-1,1,-1]

rodovia municipal: [-1,1,1,-1]

	R_1	R_2	R_3	R_4	R_5	
J_1	[[1	-1	1	-1	1	],
J_2	[-1	1	-1	1	-1	],
J_3	[-1	1	-1	1	-1	],
J_4	[1	-1	1	-1	1	]].

A partir das tabelas de reforço dos pares de associação vamos construir a tabela de sinapses, que corresponde às ligações dos dentritos com os neurônios do cérebro humano. Estas sinapses podem ser de dois tipos: (a) sinapses excitatórias e, (b) sinapses inibitórias. Se as sinapses são excitatórias, o neurônio que recebe o estímulo, tenderá a ficar excitado e a passar o estímulo adiante. Se as sinapses são inibitórias, o neurônio receptor tenderá a ficar inibido. A tabela de sinapses será igual ao somatório das tabelas de reforço.

	R_1	R_2	R_3	R_4	R_5	
J_1	[[-1+1+1	-1+1-1	1-1+1	1-1-1	1+1+1	],
J_2	[-1-1-1	-1-1+1	1+1-1	1+1+1	1-1-1	],
J_3	[1+1-1	1+1+1	-1-1-1	-1-1+1	-1+1-1	],
J_4	[1-1+1	1-1-1	-1+1+1	-1+1-1	-1-1+1	]].

	R_1	R_2	R_3	R_4	R_5		
J_1	[[1	-1	1	-1	3],	====>	Tabela de re-
J_2	[-3	-1	1	3	-1],		forço que unem
J_3	[1	3	-3	-1	-1],		rodovias a ju-
J_4	[1	-1	1	-1	-1]].		risdição

Se apresentarmos BR-101 à rede neural, ela deverá associar esta rodovia com rodovia federal. Vejamos como:

Vamos tomar o neurônio J_1. Ele tem cinco sinapses, pois ele se liga com cinco neurônios da lista de rodovias. As eficácias das sinapses de J_1 são: 1, -1, 1, -1, 3, que são as sinapses que ligam J_1 a R_1, R_2, R_3, R_4, R_5, respectivamente.

O valor das sinapses que ligam J_1 a R_5 é 3, que tem o seguinte significado: se o neurônio R_5 estiver excitado, um sinal elétrico que vier deste neurônio será excitado. Melhor, um sinal excitatório será passado para J_1.

O efeito integral dos sinais elétricos enviados a J_1 será:

$I_1 = \sum u_i . s_i$ onde: u_i --> eficácia das sinapses

s_i --> sinais elétricos de entrada na célula, vindo de outros neurônios.

$I_1 = -1+1+1-1+3 = 3$

Os sinais que chegam a J_2, J_3, J_4 são:

$I_2 = -3+1+1+3-1 = 1$

$I_3 = -1-3-3-1-1 = -9$

$I_4 = -1+1+1-1-1 = -1$

Os estímulos que chegam a um neurônio devem ser comparados com dois patamares: um patamar positivo (+1) e outro negativo (-1).

SE estímulo > patamar positivo
ENTÃO o neurônio se excita

SE estímulo < patamar negativo
ENTÃO o neurônio se inibe

Os neurônio representativos de jurisdição recebem os seguintes estímulos:

neurônio	estímulo recebido	estado final do neurônio
J_1	3	1
J_2	1	1
J_3	-9	-1
J_4	-1	-1

Assim, quando apresentamos BR-101 à rede neural, ela associou com a jurisdição apresentada por:

[1,1,-1,-1] ---> que é o padrão de rodovia federal.

4. SISTEMA ESPECIALISTA PARA AUXÍLIO À ESCOLHA DE ESTRATÉGIAS DE CONSERVAÇÃO DE RODOVIAS NÃO PAVIMENTADAS

Este capítulo descreve uma aplicação da Inteligência Artificial na Engenharia Rodoviária, na forma de um sistema especialista que auxilia na escolha de estratégias de conservação de rodovias não pavimentadas.

4.1 Objetivos

O sistema CONSER tem por objetivo principal mostrar a vantagem da utilização de técnicas de inteligência artificial na construção de sistemas computacionais que retenham o conhecimento heurístico de engenheiros rodoviários.

CONSER (ou o desenvolvimento deste) poderia ser utilizado em: (a) treinamento de técnicos e engenheiros rodoviários com pouca experiência. A base de conhecimento do sistema foi construída com base na experiência pessoal de três especialistas na área de conservação rodoviária; (b) auxiliar a engenheiros encarregados da conservação e manutenção de rodovias, como uma segunda opinião para a resolução de problemas.

4.2 Generalidades

CONSER é um sistema computacional interativo que aconselha sobre estratégias de conservação de rodovias não pavimentadas. É desenvolvido em Prolog, e roda em micro-computadores compatíveis com PC-DOS.

CONSER interroga o usuário sobre tipo de material de jazida, problemas existentes nas rodovias, tipo de equipamento disponível, etc. Os dados obtidos são gravados na sua base de dados dinâmica, e serão utilizados para fazer inferências sobre as regras que constituem a sua base de conhecimento.

O sistema, contém 65 regras de produção. Sua base de conhecimento está em expansão para envolver além de conhecimento sobre conservação de rodovias não pavimentadas, conhecimento sobre:

(a) rodovias pavimentadas (pavimentos flexíveis e rígidos);
(b) problemas na área de construção de rodovias;
(c) calibragem de usinas de asfalto e de mistura de solos;
(d) determinação de traços de brita graduada, etc.

4.3 Aquisição do Conhecimento

A fase de aquisição do conhecimento e sua formalização com uma ferramenta adequada são as tarefas principais na construção de um sistema especialista. "Aquisição de conhecimento é a transferência e transformação de habilidades ou perícia para resolver problemas contida em alguma fonte de conhecimento para um programa" [Genaro,86]. Este conhecimento pode originar-se de manuais técnicos, livros, publicações especializadas e experiência adquirida por técnicos em áreas específicas. Para **CONSER**, as fontes de conhecimento foram manuais técnicos de Departamentos Estaduais de Estradas de Rodagem e a experiência pessoal de três engenheiros rodoviários do DER/SC (Departamento de Estradas de Rodagem de Santa Catarina), numa série de entrevistas. Durante este período, problemas corriqueiros relacionados com conservação de rodovias não pavimentadas foram propostos aos engenheiros entrevistados, e, relacionou-se as estratégias utilizadas na resolução destes problemas. A maneira própria de cada um resolver os problemas era comparada com a dos outros para checar formas diferentes de atacá-los. A aquisição do conhecimento de cada um dos engenheiros consultados foi de forma individual, devido à dificuldades de reuni-los. Neste processo, foram sendo identificados as variáveis (qualificadores) e possíveis valores que estas variáveis poderiam assumir e que seriam pertinentes aos problemas na área de conservação rodoviária. Foi seguido um importante conselho dado por Waterman [Waterman,86]: "Não seja seu próprio perito!". A função do engenheiro do conhecimento é ajudar

o especialista a formalizar seus métodos de resolução de problemas. E, esta tarefa fica facilitada quando o engenheiro do conhecimento tem experiência profissional no domínio do conhecimento. Harmon [Harmon,88], sugere que o engenheiro do conhecimento deva fazer um esforço para aprender tudo o que puder sobre o domínio e a tarefa do especialista, revendo documentos e lendo livros para familiarizar-se com o domínio do problema antes de começar o processo de interação com o especialista. Esta fase foi, portanto, facilitada pela experiência do autor deste trabalho na área de conservação rodoviária.

4.4 Base de Conhecimento

A base de conhecimento de CONSER é constituída por fatos, adicionados ao programa durante sua execução pelo usuário (denominada de base de dados dinâmica) e, por regras de produção, que contêm o conhecimento obtido dos manuais técnicos pesquisados e nas entrevistas realizadas com os especialistas consultados. Estas regras de produção são declarações do tipo **SE-ENTÃO:**

SE

⟨premissas⟩

ENTÃO

⟨conclusão⟩

formalizadas apropriadamente em Prolog. Abaixo são mostradas duas destas regras:

SE

Existe carro pipa à disposição

e é periodo de estiagem

e as condições gerais do trecho são boas

ENTÃO

Use o carro pipa diariamente para evitar o pó e o desgaste do material de revestimento.

SE

Existe rolo compactador à disposição

e o tempo está bom

e as condições gerais do trecho não são muito boas

e o material de jazida é decomposição de granito

ENTÃO

Compactar após a patrolagem para obter um melhor revestimento e maior durabilidade do mesmo.

A parte **SE** destas regras é chamada de lado esquerdo, antecedente ou corpo da regra. A parte **ENTÃO** é chamada de lado direito, conclusão ou cabeça da regra. As premissas na parte antecedente devem ser satisfeitas para que a conclusão seja considerada verdadeira. Se qualquer premissa falhar, a conclusão também falhará.

Os fatos são expressados como triplas objeto-atributo-valor, ou duplas atributo_valor, utilizando-se sentenças de Horn, para formalizar a representação dos fatos. Assim, os fatos:

Existe carro pipa à disposição.

é período de estiagem.

As condições gerais do trecho são boas.

foram representadas em Prolog, como:

```
qualificador(carro_pipa,1),
qualificador(condicoes_climaticas,3),
qualificador(condicoes_gerais,6).
```

onde:

atributos = carro_pipa, condições_climáticas, condicoes_gerais.

valores = 1, 3 e 6 para os respectivos atributos

Os valores possíveis de serem assumidos pelos atributos, foram codificados, por razões de economia de tempo de digitação e de memória. O atributo carro_pipa poderia assumir os seguintes valores:

1. sim
2. não

Condicoes_climaticas poderia assumir os valores:

1. parou de chover
2. tempo bom
3. período de estiagem
4. chovendo
5. sem chover há muito tempo
6. existe previsão de chuva

Condições_gerais poderia ter:

1. ruim
2. impossivel patrolar
3. sem conservação
4. conservaçao de emergência
5. período normal de conservação
6. trecho bom
7. alguns problemas

Material de jazida:

1. decomposição de basalto
2. argilito
3. saibro

Os valores que estes atributos podem assumir são os valores utilizados pelos engenheiros consultados. Como estes especialistas

nem sempre utilizam a mesma estruturação destes valores, adotou-se o critério de relacionar todos os valores possíveis. Assim, para o atributo condicoes_climaticas, aparecem dois valores : período de estiagem e não chove há muito tempo que são tratados como valores diferentes, apesar de aparentemente, terem o mesmo significado.

4.5 Base de Dados Dinâmica

A base de dados dinâmica é definida para permitir adicionar fatos ao programa durante sua execução. Para que isto seja possível, deve-se adicionar ao programa uma seção database, e definir os predicados nesta seção, que deve ser colocada entre as seções domains e predicates. Assim, a seção database de CONSER tem os seguintes predicados:

```
database
        xpositive(symbol)
        xpergunta(string)
        qualificador(symbol,integer)
        regra_foi_usada
        ignorar(symbol)
        sinônimo(symbol,symbol)
        palavra_desconhecida(symbol)
```

Estes predicados de database podem ser usados da mesma maneira que os demais predicados. Se durante a execução do programa

ocorrer uma match com um predicado de database ele é considerado verdadeiro. Caso contrário, ele é considerado falso (ao invés de ocorrer um erro ou uma falha). Estes predicados são adicionados ao database usando-se o predicado embutido asserta, e são removidos com o predicado embutido retract.

A primeira verificação de existência de um predicado no database ocorre na cláusula:

```
pergunta(Pergunta):-
        xpergunta(Pergunta),!.
```

Esta cláusula verifica se no database tem o predicado xpergunta("Qual o objeto em análise?"). O match falhará no início do programa porque o database está vazio. O programa fará backtracking para:

```
pergunta(Pergunta):-
        ask(Pergunta),
        verifique_resposta(Pergunta,S,X),
        processe(Pergunta,X).
```

A segunda cláusula pergunta porá o predicado xpergunta no database através da primeira premissa ask. Note-se que a primeira cláusula pergunta(Pergunta), tem um cut no seu corpo. O propósito deste cut é impedir backtracking se esta cláusula tiver sucesso, pois se a pergunta já foi formulada, não há necessidade de

reformulá-la, dando uma impressão mais agradável do sistema.

O predicado xpositive tem como argumento as palavras da sentença de resposta para cada pergunta. Quando a sentença de resposta é lida, cada palavra é colocada no database como argumento de xpositive. Em seguida a cláusula converse, verificará se dentre estas palavras estão palavras-chaves que identificam valores de qualificadores. Seja, por exemplo, a seguinte cláusula:

```
converse(Pergunta):-
    Pergunta = "Existe material fino a disposição?",
    xpositive(sim),
    assertz(qualiicador(disponibilidade_de_fino,1)),
    limpar_variáveis_entrada.
```

Se para a pergunta:

Existe material fino a disposição?

for dada uma resposta **sim**, o programa adiciona ao **database** o predicado qualificador(disponibilidade_de_fino,i) onde a variável disponibilidade_de_fino tem valor 1 (codificação para o valor sim), e retira em seguida, as palavras usadas na sentença, com o predicado limpar_variáveis_entrada.

CONSER tem um pequeno vocabulário que pode ser acrescido de sinônimos. Se para a pergunta "Existe material fino à

disposição?" ao invés de responder sim, o usuário tivesse respondido afirmativo ou positivo, o sistema entenderia que o usuário queira responder sim. Quando uma palavra que for digitada não estiver no vocabulário de CONSER, ela ficará registrada no arquivo "desco.pro", e a opção editar vocabulário do MENU PRINCIPAL permite a sua adição ao vocabulário do sistema.

Os predicados sinônimo e palavra_desconhecida vão permitir esta implementação.

O último predicado de database é regra_foi_usada. Quando ele estiver no database, significa que alguma regra do banco de conhecimento foi disparada, e a mensagem:

Todas as regras foram aplicadas.

é mostrada na tela. Caso contrário, vai aparecer no fim do programa a mensagem:

Nenhuma regra foi encontrada para os dados de entrada.

4.6 Visão Geral do Funcionamento do Sistema

Uma visão geral do funcionamento de CONSER é dada na figura 4.1:

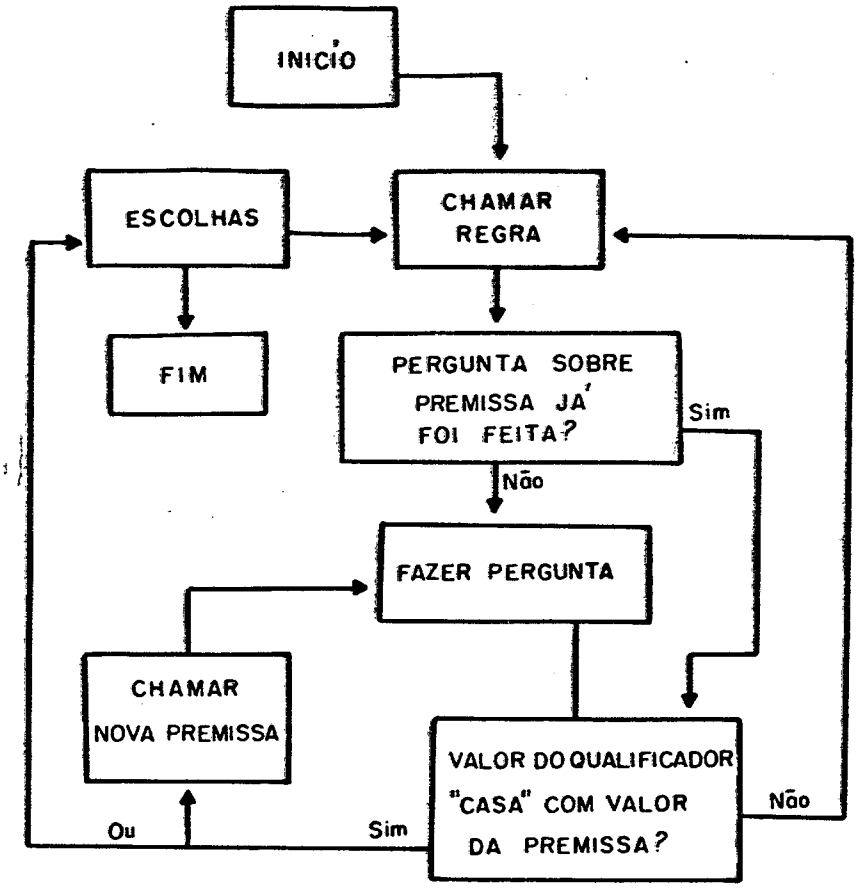


FIGURA 4.1

Na figura 4.1 **INICIO** é constituído pela parte de apresentação do sistema e pelo **MENU PRINCIPAL** que tem cinco opções:

(1) consulta sobre conservação de rodovias não pavimentadas;

(2) informações sobre o sistema;

(3) carregar vocabulário;

(4) editar vocabulário;

(5) sair do sistema.

No MENU PRINCIPAL escolhe-se a alternativa sob o cursor (digitando a tecla ENTER). Escolhendo-se a primeira alternativa, o programa vai à cláusula go_once(1), que tem a seguinte estrutura:

```
go_once(1):-
        clearwindow,
        limpe,limpe_variaveis_entrada,                (I)
        limpe_qualificadores,
        limpe_usou_regra,
        aplique_regra(_),continue,!,fail.
```

Os predicados limpe..., garantem a limpeza da base da dados dinâmica (dynamic database) antes da chamada da regra. Isto se faz necessário para o caso de uma nova consulta sem sair do sistema.

A última linha do predicado go_once(1) vai chamar todas as regras possíveis, de forma que após o disparo de cada regra, a conclusão é mostrada na tela. Isto ocorre porque o predicado aplique_regra(_) sempre falha, com exceção das duas últimas regras que tem o objetivo de verificar se alguma regra disparou ou não. Se pelo menos uma regra disparou é mostrada na tela a seguinte mensagem:

Todas as regras foram aplicadas.

caso contrário é mostrada a mensagem:

Nenhuma regra foi encontrada para os dados de entrada.

As duas últimas regras são:

```
aplique_regra2(999):-
   regra_foi_usada,                                    (II)
   write("Todas as regras foram aplicadas."),nl,nl,
   mens_fim.

aplique_regra2(1000):-                                 (III)
   not(regra_foi_usada),
   write("Nenhuma regra foi encontrada para os dados de entrada").
```

regra_foi_usada é um predicado de database que é gravado na base de dados dinâmica se alguma conclusão for mostrada na tela. Se nenhuma conclusão for encontrada, este predicado não existirá no database, de maneira que aplique_regra2(999) falhará, e aplique_regra2(1000) será verdadeiro e a segunda mensagem é mostrada ao usuário.

A premissa aplique_regra(_) da cláusula (I), vai tentar casar com a cabeça da primeira cláusula com mesmo nome, que no caso é a cláusula aplique_regra(1). A atual versão de CONSER não tem um módulo explicativo, portanto aplique_regra está sendo chamada com uma variável anônima como argumento. Para a implementação deste módulo, se faz necessário guardar o número da regra que está sendo verificada e das regras que forem disparadas, para dar explicações ao usuário quando houver solicitação para tanto. No atual estado

de CONSER não se faz necessário guardar o número das regras. A numeração foi utilizada para dar um ordenamento às mesmas.

No fluxograma da figura 4.1, passamos à fase Chamar regra. Prolog tenta casar premissas com cabeças de cláusulas de mesmo nome, de cima para baixo. Assim, o sistema tenta verificar se aplique_regra(1) é verdadeiro. Para que isto ocorra todos os predicados do corpo com a cabeça aplique_regra(1) devem ser verdadeiros. As regras do sistema CONSER tem de uma maneira geral, o seguinte formato:

```
regra(Número):-
     pergunta(<Objeto>),qualificador(<Objeto>,<Valor>),
                .
                .                                        (IV)
                .
     conclusão(<Número da conclusão>),
     fail.
```

onde o predicado pergunta(<Objeto>), verifica se a pergunta sobre <Objeto> já foi formulada. Em caso negativo, mostra a pergunta na tela e guarda a resposta no banco de dados dinâmico. Se a pergunta já foi formulada, ela não o será novamente (desta forma o programa mostra um certo nível de "inteligência"). A verificação do valor da variável guardada no database é feita pelo predicado qualificador(<Objeto>,<Valor>). Se o valor do <Objeto> dado por <Valor> na regra, casar com o valor guardado no database, o sistema passa para a próxima premissa da regra. Caso contrário, a

regra falha, e a próxima regra será verificada.

Se todas as premissas de uma regra forem verdadeiras o predicado conclusão(⟨Número da conclusão⟩) escreve na tela a conclusão que tem o número dado por ⟨Número da conclusão⟩. Adotou-se esta forma colocando-se todas as mensagens de conclusão num arquivo específico, já que se pode alcançar uma mesma conclusão por regras diferentes. No caso de CONSER uma mesma conclusão não é mostrada em mais de uma vez porque as relações das conclusões são orientadas pelos qualificadores utilizados em cada regra. Mas, um outro tipo de sistema, pode com relativa facilidade evitar a repetição de conclusões, colocando os ⟨Número de conclusão⟩ numa lista. Antes de mostrar a mensagem na tela, verifica-se se o ⟨Número de conclusão⟩ está nesta lista. Se estiver, significa que a mesma já foi mostrada ao usuário, evitando-se a repetição. Uma outra forma seria utilizar esta lista para no fim da consulta mostrar todas as conclusões numa determinada ordem, utilizando por exemplo, fatores de confiança para o ordenamento destas conclusões.

A primeira regra de CONSER é dada na cláusula seguinte:

```
aplique_regra(1):-
    problema(objeto),qualificador(objeto,4),
    problema(condições_climáticas),                    (V)
        qualificador(condiçoes_climáticas,5),
    problema(disponibilidade_de_fino),
```

```
        qualificador(disponibilidade_de_fino,1),
    escolha(1),fail.
```

A premissa problema(objeto), verifica se a pergunta:

```
        Qual o objeto em análise?
```

já foi formulada.

Prolog tenta verificar se o predicado problema(objeto) é verdadeiro, procurando uma cláusula que tenha este predicado como cabeça. Isto vai ocorre com:

```
problema(objeto):-                                    (VI)
    pergunta("Qual o objeto em análise?").
```

A premissa única pergunta("Qual o objeto em análise?"), casa com a cabeça da cláusula:

```
pergunta(Pergunta):-                                  (VII)
     xpergunta(Pergunta),!.
```

O predicado xpergunta(Pergunta) é um predicado de database. Se ele está no banco de dados dinâmico, a cláusula (VII):

```
pergunta(Pergunta):-
     xpergunta(Pergunta),!.
```

é verdadeira. A variável Pergunta está instanciada ao argumento do predicado pergunta("Qual o objeto em análise?"), ou seja:

```
Pergunta = "Qual o objeto em análise?".
```

O cut (!) não vai permitir o chamamento de outra cláusula com a cabeça pergunta(Pergunta).

O predicado de database, xpergunta(Pergunta) indica assim se o valor que Pergunta está instanciando no momento já foi mostrado na tela. Como a primeira pergunta ainda não foi formulada será chamada a cláusula:

```
pergunta(Pergunta):-
    ask(Pergunta,S),
    verifique_resposta(Pergunta,S,X),          (VIII)
    processe(Pergunta,X).
```

A primeira premissa da clásula (VIII) ask(Pergunta,S) casa com a seguinte cláusula:

```
ask(Pergunta,S):-
    write(Pergunta),nl,write(":"),              (IX)
    asserta(xpergunta(Pergunta)),
    readln(S).
```

A pergunta "Qual o objeto em análise?" será mostrada na tela, e na linha seguinte parece o prompt do usuário:

```
Qual o objeto em análise?
:
```

asserta(xpergunta(Pergunta)), coloca xpergunta("Qual o objeto em análise?") no database, e readln(S), lê a sentença de resposta do usuário. Retornando à cláusula VIII, passamos para a premissa seguinte:

```
verifique_resposta(Pergunta,S,X).
```

Este predicado casa com a cabeça da cláusula:

```
verifique_resposta(_,S,X):-
    S <> "?",
    S = X.
```

que verifica se o usuário digitou "?" para obter informações acerca da pergunta formulada. Na cláusula (VIII):

```
pergunta(Pergunta):-
    ask(Pergunta,S),
    verifique_resposta(Pergunta,S,X),
    processe(Pergunta,X).
```

a última premissa:

processe(Pergunta,X)

vai iniciar o processo de pesquisa na sentença de resposta, procurando palavras-chaves.

4.7 Interface com o Usuário

A comunicação entre o sistema e o usuário é feito em linguagem natural, utilizando-se a técnica de pegar as palavras do usuário, colocá-las numa lista, verificando-se se palavras-chaves estão nesta lista. Esta técnica, que é fácil de ser implementada, permite um diálogo agradável, sem necessidade de uso de menus. É porém uma forma de interfaceamento vulnerável à tentativas de se criar uma interpretação errada da inteligência do sistema, através da entrada de frases esdrúxulas, sem sentido, pois o sistema na atual fase não possui um parser verificador de sintaxe. O sistema porém tem algumas regras para detectar incongruências tais como jazida de basalto decomposto no litoral, etc, que alertam o usuário se este tipo de entrada ocorrer.

5. CONCLUSÕES

Especialistas na área de informática céticos em relação à Inteligência Artificial dizem que ela não passa de "fogo de palha" no mundo da informática e que não teria qualquer futuro.

Não compartilhamos deste ceticismo. Demonstramos sua aplicação na Engenharia Rodoviária, especificamente na Conservação de Rodovias. O SISTEMA CONSER poderia ser acrescentado de novas tarefas, tais como: a) determinação de traço de brita graduada; b) calibragem de usinas; c) escolha de rotas nas rodovias do Estado; d) conservação de rodovias pavimentadas.

O Brasil não é um país que possa se equiparar ao Japão e aos Estados Unidos no domínio de alta tecnologia. Desenvolvendo seu próprio campo de Inteligência Artificial, o Brasil não estará desenvolvendo algo improdutivo e de réles valor. Ao contrário, este desenvolvimento pode ajudar a conquistar o domínio da alta tecnologia e posicionar o Brasil em destaque como nação desenvolvida. Aprendendo com nossos erros, aprendemos mais, e saberemos como resolver nossos problemas específicos, que só nós poderemos resolver.

Muitos pesquisadores nas universidades brasileiras tem-se voltado para a área da Inteligência Artificial, procurando desenvolver "software". É preciso que o governo reconheça o papel estratégico que desempenham as Universidades e forneça os recursos

necessários para o setor. Veja-se os países desenvolvidos.

Na luta pela competição da liderança mundial em tecnologia de ponta, O Japão a partir de 1978, começou seu Projeto do Computador de Quinta Geração. Do total destinado ao projeto (mais de 1 bilhão de dólares), empresas particulares investiram a metade, e o governo a outra metade.

Um dos mais brilhantes cientistas na área de computação, Edward Feigenbaum, da Universidade de Stanford, acredita que o mais importante não é construir uma máquina que possa pensar. Para ele, o importante é que a humanidade está ingressando numa nova era, onde os computadores serão um instrumento revolucionário do progresso humano, deixando de serem processadores de dados e informações, para serem processadores de conhecimento.

O nosso trabalho visou principalmente formas de aplicação de IA na engenharia rodoviária. Ao invés do que se poderia supor, o computador não pode nem pretende substituir o especialista humano. O computador, poderá sim, ser um excelente consultor de quem o técnico poderá ter uma segunda opinião na solução de problemas referentes à engenharia rodoviária. Ou, no caso de urgência, ter acesso a mais informações sobre determinado tema.

6. BIBLIOGRAFIA

Alty, J.L. & Coombs, M. J. - Expert Systems: Concepts and examples, Manchester, NCC Publications, 1984.

Angulo, J. M. & Moral, A. del - Inteligencia Artificial, Madrid, Paraninfo S.A., 1985.

Araribóia, G. - Inteligência Artificial, Rio de Janeiro, Livros Técnicos e Científicos Editora Ltda, 1988.

Barr, Avron & Feigenbaum, Edward A. - The Handbook of Artificial Intelligence, Volume I, William Kaufmann Inc., 1981.

Bazarian, Jacob - Intuição Heurística: Uma análise científica da Intuição criadora, São Paulo, Editora Alfa-Omega, 1988.

Bratko, Ivan - Prolog Programming for Artificial Intelligence, Addison-Wesley Publishing Company, 1986.

Carvalho, R. Lins de - Processamento de Conhecimento, Buenos Aires, Editorial Kapelusz, 1987.

Cerqueira, Luiz Alberto & Oliva, A. - Introdução à Lógica, Rio de Janeiro, Zahar Editora S.A., 1979.

Chorafas, Dimitris N. - Sistemas Especialistas: Aplicações comerciais, São Paulo, McGraw-Hill, 1988.

Copi, Irving Marmer - Introdução à Lógica, 2a. edição, São Paulo, Editora Mestre Jou, 1978.

Cunha, Paulo Cesar Ferreira de Souza - Um sistema inteligente em LISP, Tese de Mestrado, Instituto Militar de Engenharia, Rio de Janeiro, 1985.

Ferreira, Edson de Paula - Robótica, Editorial Rapelusz S.A,Buenos Aires, 1987.

Frost, Richard - Introduction to Knowledge Base Systems, New York, Macmillan Publishing Company, 1986.

Genaro, Sérgio - Sistemas Especialistas: O conhecimento artificial, Livros Técnicos e Científicos Editora S.A., 1986.

Groover, Mikell P. et alli - Robótica: Tecnologia e programação, São Paulo, MacGraw-Hill, 1988.

Harmon, Paul & King, David - Sistemas Especialistas: a Inteligência artificial chega ao mercado, Rio de Janeiro, Campus,1988.

Hilster, David de - Futuro Promissor: Com novas tecnologias surgem produtos baseados em IA, in: PC Mundo, nov/87, no.28,vol. III.

_____ - Um sonho antigo: considerada uma área nova, a IA tem mais de 30 anos, in: PC Mundo, jun/87,no.23,vol.III.

_____ - Homem/máquina: especialistas somam esforços em busca do conhecimento, in: PC Mundo, jul/87,no.24,volIII.

_____ - Conversa natural entre micros e mainframes, in: PC Mundo, jul/88,no.36,vol. IV.

Levine, Robert I. et alli - Inteligência Artificial e Sistemas Especialistas, São Paulo, MaGraw-Hill, 1988.

Lorenzoni, Evandro - O Pater e os tributos do pioneirismo, in PC Mundo, nov/88,no.40,vol.IV.

Lucena, Carlos - Inteligência Artificial e Engenharia de Software, Rio de Janeiro, Jorge Zahar Editor, 1987.

Martins, João Pavão et alli - Ferramentas de programação em Inteligência Artificial, in: Revista de Informática, Lisboa, no.3, vol.6,maio/87.

MIKRO - revista da Publitron, Publicações Técnicas Ltda, São Paulo, fev/89.

Nievola, Júlio Cesar - Sistema especialista para auxílio ao diagnóstico médico de icterícia, Tese de Mestrado, UFSC, 1988.

Nivette, Joseph - Princípios de Gramática Gerativa, São Paulo, Pioneira, 1975.

Perini, Mário A. - A Gramática Gerativa: introdução ao estudo da grmática portuguesa, 2a. edição, Belo Horizonte, Editora Vigília, 1985.

Popov, E.P. & Yurevich, E.I. - Robotics, Moscow, Mir Publisters, 1987.

Ribeiro, Horácio da Cunha e Sousa - Introdução aos Sistemas Especialistas, LTC-Livros Técnicos e Científicos Editora S.A., 1987.

Rich, Elaine -Inteligência Artificial, São Paulo, McGraw-Hill, 1988.

Roberts, Ralph - Turbo Prolog, Rio de Janeiro, Livros Técnicos e Científicos Editora S.A., 1988.

Santos, Sylvio Silveira - Inteligência Artificial: mito ou realidade, in: Fundação JP, Belo Horizonte, no. 3,4, mar/abri/85.

Schildt, Herbert - Advanced Turbo Prolog: Version 1.1, Borland-Osborne/McGraw-Hill, 1987.

Siqueira, Ethevaldo - A Sociedade Inteligente, São Paulo, Bandeirante Editora, 1987.

Townsend, Carl - Mastering Expert Systems with Turbo Prolog, Howard W. Sams & Co., Indianópolis, 1987.

Ullrich, Roberto A. - Robótica: uma introdução, Rio de Janeiro, Campus, 1987.

Waterman, Donald A. - A Guide of Expert Systems, Addison-Wesley Publishing Company, 1986.

Weiss, Sholom M. at elli - Guia prático para projetar sistemas especialistas, Rio de Janeiro, LTC - Livros Técnicos e Científicos Editora S.A., 1988.

Winston, Patrick H. - Inteligência Artificial, Rio de Janeiro, Livros Técnicos e Científicos Editora Ltda, 1988.